UNIVERSITETET I OSLO
Institutt for informatikk

WirelessHART

Gjennomgang og implementering

Masteroppgave

60 studiepoeng

Håvard Tegelsrud
og
Jørgen Frøysadal

3. mai 2010

Forord

Denne rapporten er utført som en 60 studiepoengs masteroppgave ved Institutt for informatikk, UiO. Arbeidet er utført av studentene Jørgen Frøysadal og Håvard Tegelsrud i tidsrommet august 2009 til mai 2010, under veiledning av professor Stein Gjessing.

Vi ønsker å rette en stor takk til Stein for å ha viet oppgaven mye tid og oppmerksomhet. Allerede før semesteret startet avtalte vi ukentlige møter med Stein hvor vi diskuterte muligheter og etter hvert hvordan stoffet så ut til å bli. Stein har vist en interesse for å lære noe om et område som var relativt nytt også for han. Da stoff fra oppgaven ble levert for gjennomsyn gikk alltid Stein grundig til verks, både hva språk og innhold gjaldt. Takk til Stein for hans interesse, engasjement og sjenerøsitet!

Vi vil også takke Simon Carlsen og Stig Petersen. Våre eksterne veiledere ved Statoil var tidlig med på å diskutere problemstilling, uten å være låst til hvilke eksplisitte behov Statoil måtte ha. Hver gang vi leverte materiale til gjennomsyn fikk vi detaljerte og konstruktive tilbakemeldinger. Vi setter også pris på at Simon og Stig tok seg tid til å reise Norge på langs for et personlig møte og oppdatering på hvordan vi lå an. Takk til Simon og Stig for god kvalitetssikring og inspirasjon!

Til slutt vil vi rette en takk til Niels Aakvaag, senior systemarkitekt ved Enfo Broadcast, og nå også ansatt som forsker ved Sintef. Helt i starten av oppgaven, da vi fremdeles ikke hadde oversikt over hva vi hadde begitt oss ut på, kom vi i kontakt med Niels, som har lang erfaring i trådløse nettverk samt et glødende engasjement for hva han driver med. Gjennom hele oppgaven har Niels fulgt utviklingen og kommet med inspirerende og entusiastiske tilbakemeldinger. Takk til Niels for hans entusiasme og omfattende kunnskap!

Sammendrag

Denne rapporten gir en gjennomgang av den trådløse sensornettverkstandarden WirelessHART, samt implementeringen av det fysiske laget og store deler av datalinklaget på mikrokontrollerbrikken AVR RAVEN fra Atmel. Prosjektet rapporten tar utgangspunkt i er et samarbeid mellom Institutt for Informatikk ved Universitetet i Oslo og Statoil ASA. Statoil har mye erfaring med HART, den trådbaserte forløperen til den trådløse sensornettverkprotokollen WirelessHART, og de har også etter hvert gjort mye forskning på WirelessHART med tanke på å kunne ta det i bruk i produksjon.

Sensornettverk har vært i bruk i industrien i mange år som trådbaserte løsninger, men å etablere nye trådbaserte løsninger kan være en kostbar affære og hemmer fleksibiliteten i plasseringen av sensorene. Mye spenning har derfor vært knyttet til om nye trådløse standarder for sensornettverk, som ZigBee, WirelessHART og ISA100, når opp til kravene industrien stiller. WirelessHART ser ut til å være en god kandidat til å lykkes, og en fungerende implementering utført av Dust Networks, samt en hel del utstyr basert på denne, har vært på markedet en stund allerede. Det er imidlertid ønskelig at flere implementeringer kommer på markedet, og denne rapporten, sammen med vår implementering av WirelessHART, gir en innføring i hvordan WirelessHART kan implementeres på en brikke som er utviklet primært for ZigBee.

Rapporten gir en innføring i trådløse sensornettverk generelt, og ser på noen av de mest sentrale teknikkene og standardene som benyttes til dette. Leseren får dermed et innblikk i et fagområde som må sies å være relativt ferskt. Deretter gir vi en grundig gjennomgang og tolkning av WirelessHART-standarden spesielt. Med unntak av standarden i seg selv finnes det lite litteratur om denne fra før. Derfor, i kombinasjon med at en god forståelse av standarden er helt avgjørende for implementeringsarbeidet, har vi lagt mye vekt på denne delen. Vi har sett på de ulike algoritmene, teknikkene og konseptene som standarden legger frem, og beskrevet dem på en, i våre øyne, oversiktlig og forståelig måte.

En annen, sentral del av rapporten er implementeringsarbeidet vi har gjort, og vi åpner dette med å gi en gjennomgang av utviklingsmiljøet vi har forholdt oss til. Både for å dokumentere at brikken vi jobber på egner seg til utvikling av WirelessHART, og for å legge til rette for at andre enkelt kan føre utviklingen videre. Deretter gir vi en gjennomgang av vår

implementering, med beskrivelse og diskusjon rundt mange av de sentrale delene av vår egen arkitektur og design, herunder noen foreslåtte algoritmer i pseudokode.
Avslutningsvis legger vi frem en evaluering av de delene av protokollen vi har implementert, der vi viser at vår implementering tilfredsstiller de kravene standarden stiller, samtidig som vi diskuterer noen fremtidige utfordringer som kan dukke opp i en videre implementering.
Teksten er et resultat av en tidsbegrenset oppgave, og det er på det rene at det er rom for å arbeide videre på vår implementering. Vi avslutter derfor med en konklusjon der vi blant annet konkluderer med at WirelessHART ser ut til å være mulig å implementere på Atmel-brikken vi har benyttet, opprinnelig laget primært for andre sensornettverkstandarder, og at grunnlaget for en slik implementering er lagt gjennom arbeidet vi har gjort. Avslutningsvis presenterer vi også noen tanker vi har gjort oss rundt mulige videre arbeider på vår implementering.

God lesing.

Innhold

1 Innledning

1.1 Statoil

Prosjektet som denne teksten baserer seg på har vært et samarbeid mellom Institutt for Informatikk ved Universitetet i Oslo og olje- og gasselskapet Statoil ASA.

Som en av de største aktørene innen olje og gass har Statoil mye erfaring med HART, den trådbaserte forløperen til WirelessHART. Deres naturgassanlegg «Ormen lange» ble tildelt prisen for «HART plant of the year» i 2007, med et trådbasert sensornettverk med 2 193 km kabel og omtrent 2 200 HART-instrumenter. Med en meterpris på kabel ferdig trukket på ca 75 000 NOK er det forståelig at selskapet undersøker mulighetene for trådløse løsninger, og mye tid og forskningsressurser har vært brukt på WirelessHART og tilsvarende protokoller.

Blant annet har Statoil vært en av pionerene i arbeidet med å sette WirelessHART-nettverk på prøve i virkelige industrimiljøer. Hvis WirelessHART skal tas i bruk i stor skala, er det imidlertid i Statoils interesse at flere utstyrsleverandører av brikker og protokollimplementeringer kommer på banen, både for å skape en konkurransesituasjon som kan bidra til å holde prisene nede, og for å få noder som varierer i visse egenskaper som gjør det mulig å velge node etter hvilke behov det er ønskelig å dekke.

1.2 Problemstilling

Etter at WirelessHART ble lansert i 2007 tok det et års tid før DUST Networks etablerte seg som den eneste leverandøren av WirelessHART-kompatibel maskin- og programvare mot slutten av 2008 [1]. Disse brikkene består av alt det nødvendige for å virke i et WirelessHART-nettverk, inkludert radio, prosessor og implementering av protokollen. Leverandører av sensorutstyr til industrien monterer så sine egne antenneforsterkere og legger til sitt eget applikasjonslag i tråd med HART-protokollen.

WirelessHART er en ung teknologi, og det er ønskelig at det skal komme flere leverandører på markedet, både av radiobrikker og protokollimplementeringer. Leverandører som leverer utstyr til sluttbrukeren (eks. Siemens, Emerson og Endress+Hausser, leverandører av komplette WirelessHART-sensornoder) er interessert i dette da det vil holde prisen nede på utstyr fra underleverandører. Men også Hart Communication Foundation (HCF) og deres medlemmer ønsker en lav produksjonskostnad for å holde liv i protokollen [2]. I tillegg vil

forskjeller mellom maskinvare kunne spille inn på ytelse (for eksempel effektforbruk og RF-sensitivitet). I et marked med variasjon i noders egenskaper vil en sluttbruker kunne velge ut den typen som passer best til sitt spesielle bruk. For ZigBee, en annen standard for trådløse sensornettverk, ser man flere eksempler på ytelsesforskjell mellom noder fra forskjellige leverandører: ZigBee-radioen CC2420 fra Texas Instruments har en sensitivitet på -95 dBm og en maksimal sendestyrke på 0 dBm [3], mens ZigBee-radioen AT86RF230 fra Atmel har en sensitivitet på -101 dBm og en maksimal sendestyrke på 3 dBm [4]. Disse forskjellene er med på å skape variasjoner blant tilgjengelige sensornoder på markedet, noe man også ønsker for WirelessHART.

Enn så lenge domineres markedet av DUST Network sine enheter, men i 2009 [5] ble det på en messe i Frankfurt demonstrert en enhet fra et annet firma, STG (Software Technologies Group). I tillegg mener selskapet NIVIS at de også har en enhet som vil fungere i et WirelessHART-nettverk sammen med noder fra andre leverandører, og som har spennende muligheter i konvergensen mellom WirelessHART og ISA100 [6]. Dette er en positiv utvikling, men det gjenstår å se om noen av disse vil bli tatt i bruk i kommersielle enheter.

Trådløse sensornettverk er et viktig og voksende marked, og vi har beskrevet mange gode grunner til å ønske flere leverandører av WirelessHART-brikker, slik det finnes for for eksempel ZigBee. Siden brikkene laget for ZigBee vanligvis har bedre sensitivitet enn brikken fra DUST Networks (som alle eksisterende WirelessHART-noder benytter), har vi satt oss som mål å implementere WirelessHART på en slik enhet. Dette vil gjøre det enklere og raskere å få flere aktører på banen, siden det i så fall ikke vil bli nødvendig med ny maskinvareutvikling. Som vi skal se tyder vår påbegynte implementering og etterfølgende evaluering på at dette lar seg gjøre.

1.3 Atmel

I oppstartsfasen og så lenge vi ikke visste tilstrekkelig om verken IEEE 802.15.4-standarden eller WirelessHART var vi på utkikk etter et utviklingsmiljø. Vi hadde behov for både maskinvare og programvare, og kom i den forbindelse i kontakt med norgeskontoret til Atmel, som var villige til å bistå med alt vi måtte ha behov for.

Atmel produserer mikrokontrollere og flashminne, og fokuserer på utvikling av systemer hvor alt er samlet på ett kort («system-level solution»). Deres serie med mikrokontrollere som kom i 1996, og går under betegnelsen AVR, var blant de første mikrokontrollere som benyttet flashminne på brikken for lagring av programmet som kjører på brikken. (Med bruk av

flashminne kunne man programmere brikken gjentatte ganger, i motsetning til tidligere teknologi hvor man måtte programmere brikken en gang for alle). Brikkene til Atmel har god sensitivitet sammenlignet med for eksempel en tilsvarende brikke fra Texas Instruments [3], og også om man sammenligner med DUST Networks WirelessHART-brikke [7]. Vi mener det vil være interessant å se en fullverdig WirelessHART-protokoll på en brikke fra Atmel som opprinnelig er designet primært for ZigBee og IEEE 802.15.4-trafikk, og som har bedre sensitivitet enn den eksisterende WirelessHART-brikken.

1.4 Oppgavebeskrivelse

Ut fra det vi har diskutert så langt kan vi nå snevre oss inn til en konkret oppgavebeskrivelse. Det endelige målet er å implementere de det fysiske laget og datalinklaget i WirelessHART samt gi en vurdering av hvorvidt WirelessHART lar seg implementere på en ZigBee-tilpasset brikke fra Atmel. Om dette lar seg gjøre vil vår implementering kunne videreutvikles til en komplett implementering av WirelessHART. Oppgaven må, slik vi ser det, utføres i to steg.

Første steg er å forstå WirelessHART-standarden, både med tanke på å finne ut hvilke fysiske krav som stilles til brikken, og for å kartlegge hvordan en implementering bør designes. WirelessHART-standarden er relativt omfattende, samtidig som den legger til rette for at utvikleren selv tar mange av de implementeringsspesifikke avgjørelsene. En grundig gjennomgang av standarden er derfor nødvendig for å forstå valgene vi har gjort i vår egen implementering av WirelessHART. I tillegg er WirelessHART, sannsynligvis grunnet sin unge alder, forholdsvis lite dokumentert og diskutert andre steder, og derfor anser vi en slik gjennomgang og tolkning av WirelessHART-standarden som en viktig del av oppgaven.

Andre steg er å vurdere Atmel-brikkens egnethet som maskinvare for en WirelessHART-implementering ved å se hvorvidt vi klarer å holde oss innenfor de krav standarden stiller. Dette gjøres til en viss grad ved å sammenlignene de fysiske kravene i standarden mot hva Atmel spesifiserer at deres brikke kan levere. Hovedvekten av arbeidet ligger imidlertid i å implementere, teste og evaluere noen av de mest sentrale elementene i protokollen. Ved å dokumentere dette underveis, vil vi samtidig legge grunnlaget for en komplett implementering av WirelessHART på et senere tidspunkt. Store deler av denne teksten er altså en rapport av vår påbegynte WirelessHART-implementering.

1.5 Rapportens struktur

Før leseren begynner på teksten er det på sin plass å redegjøre for hvordan den er bygget opp.

Teksten kan sees på som todelt, der kapittel én til fire utgjør den teoretiske delen og de resterende kapitler den mer praktiske delen. Vi innleder med et kapittel som klargjør problemstillingen samt gir en introduksjon til involverte parter i prosjektet. Videre settes WirelessHART i en kontekst, ved at vi gir en del bakgrunnsinformasjon om sensornettverk generelt og noen viktige trådløse sensornettverksprotokoller spesielt. WirelessHART-standarden blir selvsagt grundig gjennomgått, og vi presenterer vår tolkning av de sentrale delene av denne standarden, først og fremst på det fysiske laget og datalinklaget.

Vi går så over i den mer praktiske gjennomgangen med å beskrive utviklingsmiljøet vi har forholdt oss til, og deretter går vi gjennom vår egen implementering med beskrivelse og diskusjon rundt noen sentrale deler av vår egen arkitektur og design. Avslutningsvis har vi også en evaluering av de delene av protokollen vi har implementert, der vi viser at vår implementering tilfredsstiller de kravene standarden stiller.

Teksten er stort sett skrevet «nedenfra og opp» i den forstand at når vi går gjennom en protokoll tar vi først for oss det fysiske laget for så å bevege oss oppover til datalinklaget og så videre. Ett unntak som er verdt å merke seg er i kapitlet om datalinklaget i WirelessHART, hvor vi forklarer sublagene i datalinklaget ovenfra og ned.

1.6 Definisjoner

Begrepene «node» og «enhet» benyttes begge om maskinvare med en kjørende nettverksprotokoll, og har således samme betydning og brukes om hverandre i denne teksten. Begrepet «brikke» benyttes for å gi en mer spesifikk distinksjon mellom selve brikken som kjører protokollen og enheten som brikken er bygget inn i. For eksempel har produsentetn Siemens laget WirelessHART-enheter som er bygget rundt DUST Networks WirelessHART-brikker.

Av mer tekniske definisjoner har vi valgt å spare disse til kapittel 3.1 om nettverksteori. For øvrig henviser vi til ordlisten bakerst i dokumentet hvor en del sentrale begrep og forkortelser er beskrevet.

2 Introduksjon til trådløse sensornettverk

Vårt arbeid dreier seg i all hovedsak om trådløse sensornettverk, og noen av de vanligste variantene av disse: ZigBee og ZigBee Pro, ISA100 og ikke minst WirelessHART, samt deres felles utgangspunkt IEEE 802.15.4-standarden. I dette kapitlet tar vi for oss trådløse sensornettverk på et generelt grunnlag, før vi i det neste kapitlet vil ta for oss de aktuelle protokollstandardene mer spesielt, og forsøke å danne et inntrykk av hvordan de er bygget opp og er relatert til hverandre.

2.1 Trådløse personlige datanettverk

WLAN og trådløse nettverk er i dag kjente begreper, også blant ikke-teknologer. Standarden som brukes i typiske hjemmenettverk, på arbeidsplasser, skoler, kafeer og lignende, ble første gang utgitt i 1997 og har siden kommet i en rekke varianter. Den opprinnelige standarden het IEEE 802.11, men er nå utdatert. I stedet er det varianter som eksempelvis IEEE 802.11b og IEEE 802.11g som har overtatt.

Blant hovedkravene til WLAN er lengst mulig rekkevidde, muligheten til å sømløst kunne skifte tilgangspunkt, samt at protokollen har kapasitet til å behandle mange samtidige enheter [8]. I mange tilfeller har man imidlertid ikke disse behovene, men ønsker bare trådløs dataoverføring mellom enheter i det umiddelbare nærområdet. Eksempler på dette kan være trådløs overføring mellom et tastatur og datamaskinen, eller mellom en bærbar musikkspiller og trådløse hodetelefoner. Slike nettverk kalles WPAN (Wireless Personal Area Network), eller trådløse personlige datanettverk på norsk. Her kreves det først og fremst at nettverket skal fungere i det nære arbeidsområdet til en person, satt til å være omkring 10 meter i alle retninger. IEEE har standardisert tre klasser av slike trådløse personlige datanettverk ut fra hva de kan tilby av egenskaper. Den mest kjente av disse er nok IEEE 802.15.1, som var identisk med Bluetooth 1.1 ved ratifiseringen i 2002 og ble oppdatert til Bluetooth 1.2 i 2005, og som derfor ofte bare omtales som Bluetooth-standarden. Produsentorganisasjonen Bluetooth Special Interest Group har imidlertid videreutviklet standarden til versjon 2 og 3 uten at IEEE har ratifisert disse, og IEEE 802.15.1 og Bluetooth kan dermed ikke lenger sies å være det samme. IEEE 802.15.1-nettverk har middels hastighet og tilbyr en QoS (quality of service, tjenestekvalitet) som egner seg til for eksempel tale, slik vi kjenner det fra trådløse

hodetelefoner til mobilen. Videre finnes IEEE 802.15.3 som har svært god hastighet og QoS, og derfor er godt egnet til multimedia-applikasjoner. Ulempen med denne teknologien er et høyere strømforbruk enn IEEE 802.15.1.

Sist men ikke minst har vi altså IEEE 802.15.4, som tilbyr vesentlig lavere hastigheter enn de andre to standardene, men som dermed også har et vesentlig lavere strømforbruk. IEEE 802.15.4 kalles av denne grunn for et Low-Rate WPAN. LR-WPAN brukes der vanlig WPAN blir for kostbart, der man trenger svært lavt strømforbruk, eller der man rett og slett ikke har behov for ytelsesforbedringene som for eksempel IEEE 802.15.1 kan tilby.

I prinsippet kan IEEE 802.15.4-standarden brukes til ulike formål, men IEEE-arbeidsgruppen som utviklet standarden gjorde det med instruks om å fokusere på trådløse sensornettverk. IEEE 802.15.4 er med andre ord forsøkt tilpasset denne type nettverk spesielt [8]. I Tabell 2-1 vises de ulike egenskapene til IEEE 802.15.4 sammenlignet med IEEE 802.11b og IEEE 802.15.1. Rekkevidden på IEEE 802.15.4 er satt til 10 meter, men i praksis vil den normalt kunne ventes å være nærmere 50 meter. (Dersom man når 10 meter med -85dBm sensitivitet, kan man ved fri sikt nå 40 meter med -97dBm (rekkevidden dobles for hver 6dB), en typisk ytelse på kommersielle radioer).

	WLAN (IEEE 802.11b)	**Bluetooth 1.2 WPAN (IEEE 802.15.1)**	**LR-WPAN (IEEE 802.15.4)**
Rekkevidde	~100 meter	~10-100 meter	10 meter
Overføringshastighet	~2-11 Mb/s	1 Mb/s	< 0,25 Mb/s
Strømforbruk	Middels	Lavt	Ultralavt
Størrelse	Stor	Mindre	Minst
Kostnad/kompleksitet	Høy	Middels	Lav

Tabell 2-1 IEEE 802.15.4 sammenlignet med andre trådløse teknologier [8]

2.2 Trådløse sensornettverk

Trådløse sensornettverk er en spesiell type nettverk for trådløs overføring av data til, fra og mellom sensorer og aktuatorer. Grunnlaget for fremveksten av denne typen nettverk skyldes den rivende utviklingen innen mikroelektronikk de siste 50 årene. Ikke bare har antall transistorer integrert på en enkelt brikke økt eksponentielt, prisnivået på brikkene har også sunket til en brøkdel. I 1965 var man i stand til å produsere integrerte kretser med 50 transistorer, og på den tiden kostet transistorene 30-40 kroner hver. I 2006 hadde vi passert en

milliard transistorer på en integrert krets, og prisen for hver enkelt av dem lå på omkring 0,000002 øre. Dette er billigere enn et enkelt riskorn, og med en årsproduksjon på over en million billioner transistorer må riskorn også se seg slått på antall [9].

Denne utviklingen har ført til en ny rolle for datamaskiner. Omkring en gang i tiåret dukker det i følge Culler et. al. en ny klasse datamaskiner opp [10], og utviklingen har tatt oss fra mainframes via minidatamaskiner og den personlige datamaskinen til mobile datamaskiner. Hver av dem har vært avhengig av teknologisk utvikling som har tillatt en ny formfaktor, og hver av dem har ført til nye bruksområder for datamaskinteknologi. Culler et. al. mener at trådløse sensornettverk kan være en ny slik klasse, siden de følger trenden i størrelse, antall og pris. De skiller seg imidlertid kraftig fra de andre klassene hva bruksområde angår.

2.2.1 Nettverksnodene i trådløse sensornettverk

Trådløse sensornettverk består av små noder, hver enkelt utstyrt med sensor(er), kommunikasjonsradio, og en viss datakraft til å måle og analysere data. Et viktig poeng ved slike nettverk er deres evne til å administrere og organisere seg selv (Ad-Hoc) og ikke trenge noen spesifikk infrastruktur. Selv om enhetene i seg selv er små og billige, blir de straks langt mindre interessante hvis det blir nødvendig med masse arbeid, kabling og testing for å iverksette dem, eller i forbindelse med rekonfigurering ved mangelfull kontakt, ikke-fungerende noder og lignende. Dette er i stor grad tilfelle ved trådbaserte sensornettverk, og har vært en av hovedmotivasjonene bak fremveksten av trådløse alternativer [8]. «Smart Dust»-prosjektet som pågikk på Berkley-universitetet rundt årtusenskiftet så for seg nettverk så fleksible at nodene mer eller mindre kunne «kastes» inn i for eksempel et katastrofeområde, og deretter konfigurere seg selv for å kunne gi nyttig kartleggingsinformasjon til personell på vei inn [11]. For at dette skal være mulig må de trådløse nodene ha datakraft nok til relativt sofistikerte nettverksalgoritmer, samtidig som de må kunne tilby akseptabel overføringshastighet og radiorekkevidde. I tillegg stilles det høye krav til både størrelse, effektforbruk og pris. Effektforbruk er særlig viktig i og med at nodene er trådløse og derfor avhengige av å driftes av et batteri eller energihøsting (f.eks. solcellepanel). Med et potensielt stort antall noder er det avgjørende å ha lite vedlikehold forbundet med nettverket, og når det i mange tilfeller også kan være fysisk vanskelig eller umulig å komme til nodene etter at de er utplassert, forstår vi at det etterstrebes å få batteriene til å leve så lenge som mulig. Nodene bruker radiofrekvens (RF) som kommunikasjonsmiddel siden denne teknologien ikke er like avhengig av fri sikt som for eksempel infrarød-teknologi

(IR), og siden den kan skaleres etter behov med tanke på effektforbruk vs. hastighet og rekkevidde [8]. Likevel, RF-senderen er et strømsluk i forhold til resten av noden, og strømforbruket kan reduseres med over 90 % med radiosenderen avslått. Slår man i tillegg av mikrokontrolleren, og dermed også selve sensoren, kan man oppnå en reduksjon i effektforbruk på over 99 % [12]. En rask utregning viser at et standard AAA-batteri (750 mAh) kan vare i to år på vanlige radiosendere (10 mA), forutsatt at man har en aktivitetstid på mindre enn 0,5 %, og i over fire år med en aktivitetstid på 0,1 %. [8]. Det eneste man da har aktivt er sanntidsklokka som holder styr på tiden, slik at noden kan reaktivere seg selv på gitte, predefinerte tidspunkt for å utføre målinger eller overføre data. I et nettverk bestående av sovende noder blir det nødvendigvis viktig med synkronisering, siden nodene må sove og være aktive på samme tidspunkt. Dette kan, som vi skal se mer i detalj senere, løses ved hjelp av diverse nettverksprosedyrer.

2.2.2 Trådløse sensornettverk i praksis

Til nå har vi konsentrert oss om de tekniske aspektene ved trådløse sensornettverk, men et annet spørsmål er hva de kan brukes til. Culler et. al [10] beskriver en tredeling i bruksområder; overvåking av rom, overvåking av gjenstander, og overvåking av interaksjonen mellom gjenstander (og nærliggende rom). Med overvåking av rom er det lett å tenke på innbruddsdetektering, men andre eksempler er overvåking av et miljø som mikroklimaet i et stort tre eller i forbindelse med jordbruk [8]. Her er sensorer for blant annet sollys, temperatur, fuktighet og barometrisk trykk aktuelle å ta i bruk, hvor dataene så prosesseres og rutes gjennom nettverket av noder og ut til verden utenfor.

I den andre kategorien, overvåkingen av gjenstander, vil det typisk være snakk om overvåking av utstyr og strukturer i industrien, men det kan også omfatte blant annet medisinsk diagnostikk. Den siste kategorien, overvåking av interaksjonen mellom gjenstander og overvåking av interaksjonen mellom gjenstander og nærliggende rom, dreier seg om mer komplekse løsninger, som for eksempel håndtering av store katastrofer, produksjonsprosessflyt og overvåking av dyreliv.

2.3 Trådløse sensornettverk i industrien

Blant de nevnte eksemplene til Culler et. al. [10] finner vi overvåking av produksjonsprosessflyt og overvåking av utstyr og strukturer i industrien som bruksområder

for trådløse sensornettverk. Industrien har i lang tid benyttet seg av sensornettverk til slike oppgaver, men da kun trådbaserte. Dette har typisk vært enkle, analoge grensesnitt som 0-5 V- eller 4-20 mA-sløyfer. Argumentene for å ta i bruk trådløse løsninger er imidlertid flere, og vi har allerede vært innom et par av dem: selvorganiserende nettverk som trenger lite konfigurering, lite eller ingen kabling, samt enklere håndtering av ikke-fungerende noder. Trådbasert nettverk har høy installeringskostnad på grunn av kabling, og det er også store utgifter knyttet til vedlikehold grunnet lav skalerbarhet og høy feilrate på tilkoblinger. I tillegg fører trådløs teknologi med seg andre fordeler som støtte for mobilitet og mulighet for enkelt å innføre redundans i nettverket, samt at enhetene kan plasseres på mer utilgjengelige steder. Kravene til et trådløst sensornettverk er imidlertid noe annerledes i industrien enn de er i en del andre sammenhenger. Flammini et. al. [13] nevner blant annet sensorstørrelse som lite relevant for industrien. Dette har vært et av fokusområdene i arbeidet med trådløse sensornettverk generelt, men er i de fleste tilfeller ikke veldig viktig for industrielt bruk. I stedet trekkes robusthet frem som en nøkkelfaktor, siden nodene ofte vil bli plassert i barske omgivelser. Petersen et. al. [14] har identifisert følgende hovedkrav som viktige for olje- og gassindustrien, hvorav de fleste eller alle sannsynligvis vil være gjeldene også i andre industrier:

- *Lang batteritid*. Hvis antallet sensorer blir høyt, og noen i tillegg er vanskelig tilgjengelige, kan kort batteritid føre til mye uønsket utskiftingsarbeid.
- *Kvantifiserbar nettverksytelse*. Personell og utstyr i bevegelse og endringer i temperatur og luftfuktighet kan påvirke overføringskvaliteten. Det trådløse sensornettverket må ha mekanismer som motvirke dette, og gi stabilitet og tilgjengelighet på nivå med trådbaserte nettverk.
- *Koeksistens med WLAN*. Det trådløse sensornettverket må fungere problemfritt sammen med WLAN – altså ingen merkbar endring i ytelse hos noen av nettverkene når de opererer i samme område.
- *Sikkerhet*. Data som sendes gjennom luften er mer utsatt for sikkerhetstrusler enn data over tråd. Derfor må det trådløse sensornettverket tilby sikkerhet mot avlytting og mot avbruddsangrep (denial of service).
- *Åpne, standardiserte systemer*. Dette gir industrien mulighet til å velge fritt mellom tilbydere av utstyr, med trygghet om at de vil fungere sammen, og livstiden til teknologien blir dermed ofte lenger. Riktignok er slike løsninger ofte sene til å nå

markedet, og sikkerhetsmekanismene blir offentlige og på den måten noe mer sårbare for angrep, men dette veies likevel opp av fordelene.

Det finnes flere løsninger som helt eller delvis oppfyller disse kravene, og som derfor er aktuelle i industriell sammenheng. Vi skal i det neste kapittelet se nærmere på det som virker å være tre av de mest lovende standardene, nemlig ZigBee, ISA100.11a og ikke minst WirelessHART.

3 Protokoller for trådløse sensornettverk

Vi har allerede vært inne på at det finnes flere protokoller for trådløse sensornettverk, og at ZigBee, ISA100.11a og WirelessHART virker å være de mest lovende av disse for industrielt bruk. Siden alle tre er relativt ferske standarder (første utgave av ZigBee kom i 2004, WirelessHART i 2007 og ISA100 så sent som september 2009) er det vanskelig å si noe sikkert om utbredelse og hvilken protokoll som eventuelt vil bli den dominerende. Det vi imidlertid *kan* si noe om er de ulike protokollenes oppbygning, hva som skiller dem, og hvilke relevante egenskaper de har eller mangler, samt litt mer om hvorfor vi har valgt å fokusere på WirelessHART videre. I denne omgang nøyer vi oss med en nokså overordnet beskrivelse av de ulike protokollene, for så senere å komme tilbake til en mer grundig gjennomgang av WirelessHART i kapittel 0.

Før vi ser på de tre protokollstandardene vil vi gå gjennom noen av de mest sentrale tekniske konsepter og teknikker vi vil møte på, samt en gjennomgang av IEEE-802.15.4-standarden som alle de tre protokollene i større eller mindre grad er basert på. Her vil vi særlig fokusere på det fysiske laget, siden det er denne delen WirelessHART implementerer..

3.1 Nettverksteori

Nettverk som opererer på et delt medium har noen utfordringer de må ta hånd om. En rekke spesialiserte protokoller er utviklet for å håndtere disse utfordringene, og flere av dem er implementert i nettverksprotokollene vi skal se nærmere på. Det er derfor nyttig å ha en innsikt i hva de innebærer og hvilke utfordringer de er rettet mot. Vi vil i dette underkapittelet gi en introduksjon til de mest sentrale protokollene på MAC-laget som vi kommer til å møte igjen i senere kapitler: CSMA/CA, TDMA og kanalhopping. Aller først vil vi imidlertid se på konseptet med lagdeling av nettverksprotokoller, og hvordan dette gjenspeiles i de ulike nettverksprotokollene.

3.1.1 Lagdeling av nettverksprotokoller

ISO/OSI-modellen

Det fulle navnet på denne modellen er «Open System Interconnection Reference Model», og den er definert av International Organization for Standardization (ISO). Arbeidet med utviklingen strekker seg helt tilbake til 70-tallet, men den endelige ISO-standarden ble utgitt i 1994 [15]. Den gir en abstrakt beskrivelse av lagdelt design av nettverksprotokoller, og definerer en referansemodell for dette på syv lag. Nederst det fysiske laget og datalinklaget, deretter nettverks-, transport-, sesjons-, presentasjons- og applikasjonslaget. Et lag er i denne sammenheng en samling av konseptuelt like funksjoner som tilbyr tjenester til lag høyere opp gjennom såkalte tjenesteprimitiver (nærmere forklart senere i kapitlet), og som mottar tilsvarende tjenester fra lag lenger ned. I Tabell 3-1 ser vi en fremstilling av denne lagdelingen og hvordan dette er overført til IEEE 802-modellen, ZigBee og WirelessHART. Vi ser at protokollene ikke nødvendigvis tar i bruk alle lagene ISO/OSI-modellen definerer, men heller velger å trekke funksjonaliteten inn i de andre lagene. Dette er på ingen måte unormalt, og det samme gjøres blant annet i TCP/IP, et vanlig eksempel på en femlagsmodell uten et sesjons- eller applikasjonslag.

IEEE 802-modellen

IEEE 802 er en familie av standarder som omhandler lokaldatanett og bydatanett. Det fulle navnet på arbeidsgruppen i IEEE som holder på med dette standardiseringsarbeidet er «IEEE Local Area Network/Metropolitan Area Network Standards Committee», og herfra kommer kjente standarder som IEEE 802.3 Ethernet og IEEE 802.11 WLAN. Denne arbeidsgruppen har også utviklet standarden IEEE 802.15.4 som er tilpasset trådløse sensornettverk, og som fungerer som basis for både ZigBee-, ISA100.11a- og WirelessHART-standardene.

Alle IEEE 802-standardene er basert på samme oppbygning med utgangspunkt i ISO/OSI-modellen. IEEE 802-modellen dekker imidlertid bare de to nederste lagene, altså det fysiske laget og datalinklaget. Resten spesifiseres gjerne av industrielle konsortium av selskap som har interesse i produksjon og bruk av en gitt standard [8]. I tilfellet IEEE 802.15.4 er ZigBee og WirelessHART eksempler på dette, gjennom henholdsvis The ZigBee Alliance og The HART Communication Foundation.

Ut over bare å dekke de to nederste lagene, er en annen viktig forskjell mellom OSI-modellen og IEEE 802-modellen at datalinklaget er delt i to såkalte sublag. Det nederste kalles Media Access Control (MAC), og er spesifikt for hver enkel av IEEE 802-standardene. Det øverste

sublaget kalles Logical Link Control (LLC), altså logisk linkkontroll, og er felles på tvers av IEEE 802-standardene. Dette er derfor definert i sin egen standard, IEEE 802.2 [16]. Tabell 3-1 viser forholdet mellom ISO/OSI-modellen, IEEE 802-modellen, ZigBee og WirelessHART.

ISO/OSI-modellen	IEEE 802-modellen	ZigBee	WirelessHART
7: Applikasjon	Øvrige lag uspesifisert, og overlatt til produsenter av utstyr som bruker den underliggende standarden	ZigBee applikasjonslags-rammeverk	HART kommandoorientert applikasjonslag
6: Presentasjon			
5: Sesjon			
4: Transport			HART streamprotokoll med autosegmentering
3: Nettverk		ZigBee maske-/stjerneprotokoll	WirelessHART maskeprotokoll
2: Datalink	Logical Link Control (LLC)	IEEE 802.2 LLC	WirelessHART LLC
	Media Access Control (MAC)	IEEE 802.15.4 MAC	WirelessHART TDMA MAC
1: Fysisk	Fysisk signalering (PHY)	IEEE 802.15.4 PHY	IEEE 802.15.4 PHY

Tabell 3-1 ISO/OSI-modellen

Tjenesteprimitiver

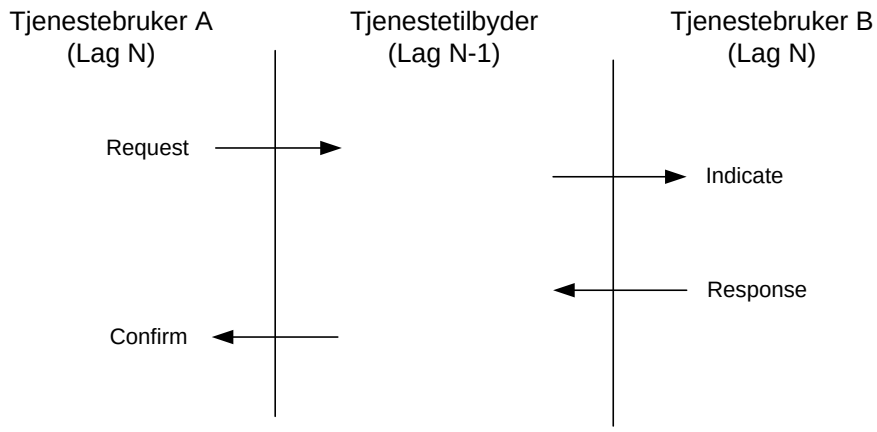
I den tidligere nevnte IEEE 802.2-standarden fra 1998 ble tjenesteprimitiver innført som et formalisert konsept til benyttelse i nettverksterminologi. Dette er et abstraksjonskonsept både IEEE 802.15.4, WirelessHART og andre protokollstandarder benytter seg av, og vi kommer derfor til å møte på det i flere sammenhenger. Av den grunn er det viktig å ha en viss forståelse for hva tjenesteprimitiver er.

Et lags tjenester defineres som å være funksjonaliteten det tilbyr til det neste overliggende lag eller sublag. Rent spesifikt utføres en slik tjeneste gjennom en viss informasjonsflyt mellom lagene, og denne flyten kan deles inn i ulike diskrete hendelser. Et tjenesteprimitiv er en

overordnet spesifikasjon av en slik hendelse; den forteller hva hendelsen skal utføre og hvilken informasjon som er påkrevet, uten å si noe konkret om implementering. 802.2-Standarden definerer fire generiske hovedtjenesteprimitiver i en slik informasjonsflyt, som sammen skal kunne kombineres til enhver tjeneste:

- *REQUEST*. Sendes fra et lag til det nærmeste underliggende lag for å initiere en tjeneste.
- *CONFIRM*. Sendes fra underliggende lag for å formidle resultatet av en eller flere tidligere *request*s.
- *INDICATION*. Sendes fra underliggende lag til laget over for å indikere en hendelse som er av betydning for laget over.
- *RESPONSE*. Sendes til underliggende lag som avslutning av en prosedyre satt i gang av en tidligere sendt *indication*.

En tjeneste kan dermed spesifiseres som tjenesteprimitivene og parameterene som tilsammen utgjør tjenesten. Hver tjeneste består av ett eller flere primitiver, og hvert primitiv kan ha flere (eller ingen) nødvendige parametere. Figuren under (Figur 3-1) illustrerer bruken av tjenesteprimitiver mot en tilbyder av en tjeneste.



Figur 3-1 Tjenesteprimitiver

Vi kan konkretisere dette ved å se på et tenkt eksempel der tjenesten som tilbys er dataoverføring. Et lag sender en *request* med dataene som parameter. På mottakersiden får det tilsvarende laget en *indicate* som inneholder de mottatte dataene som parameter, og svarer på

dette med en *response* med en parameter som forteller tjenestetilbyderen om hendelsen var vellykket eller ikke. Avsender får ut fra dette en *confirm* om at overføringen har funnet sted og hvordan det gikk. Merk at dette bare er et eksempel; en dataoverføringstjeneste kan godt ha en annen informasjonsflyt og dermed en annen bruk av tjenesteprimitiver enn vist her. Hovedsaken er at informasjonsflyten i alle tjenester, dataoverføring som andre, skal kunne defineres ved hjelp av en eller flere av tjenesteprimitivene *reques*t, *indicate*, *response* og *confirm*.

3.1.2 CSMA/CA

Trådløse sensornettverk, og trådløse nettverk generelt, sender data gjennom et delt medium, nemlig det omkringliggende luftrommet. Dette kan fort bli problematisk, for hvis to eller flere enheter sender pakker på samme frekvens på samme tidspunkt vil begge pakkene bli uleselige. En utbredt protokoll for å bøte på slike problemer i delte medier er CSMA-protokollen (Carrier Sense Multiple Access). Når en enhet har noe å sende vil den med denne protokollen først lytte til kanalen for å se om noen andre overfører data på dette tidspunktet. Hvis kanalen er ledig vil enheten sende, og hvis kanalen er opptatt vil enheten vente. I noen varianter av CSMA vil enheten i sistnevnte tilfelle fortsette å lytte til den oppfatter at kanalen er ledig, for så umiddelbart begynne å sende. Dette har den ulempen at hvis flere enheter står og venter samtidig vil alle starte sin transmisjon på nøyaktig samme tidspunkt, og vi får dermed kollisjoner og pakketap. En forbedret variant av dette er p-persistent CSMA, der pakken blir sendt med en sannsynlighet av p når kanalen blir ledig. Hvis eksempelvis p er satt til 0,5 vil pakken bli sendt av gårde i halvparten av tilfellene der kanalen er ledig. I de tilfellene den ikke blir sendt venter den en bestemt tidsperiode og gjør samme prosess på nytt. Dette reduserer kollisjonsfaren noe, og gir dermed bedre utnyttelse av nettverket [17].

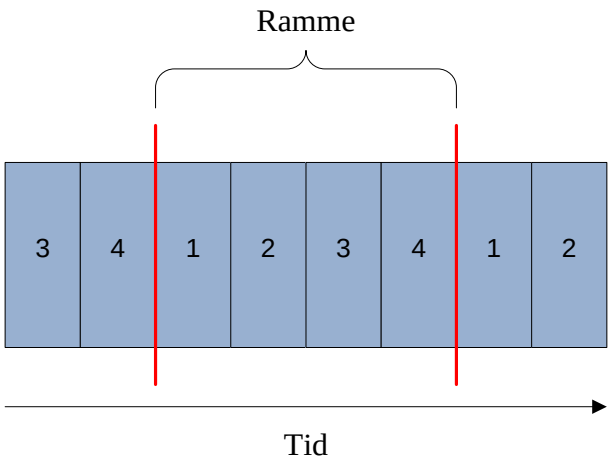
I trådbaserte nettverk brukes CSMA gjerne sammen med en kollisjonsdeteksjonsprotokoll (CD – Collision Detection). Dette betyr at hvis to noder på tross av virkemidlene i CSMA skulle ende opp med å sende data samtidig, kan de selv merke at det oppstår kollisjon på linjen og avbryte sendingen med en gang. Dataene er uansett ødelagt, så det er ikke noe poeng i å fortsette overføringen, og i stedet frigjøres kanalen umiddelbart. I trådløse nettverk har vi ikke denne muligheten, siden radioene ikke kan både sende og lytte samtidig. Derfor benytter mange trådløse nettverksprotokoller [18, 19] seg av varianten CSMA/CA i stedet (CSMA with Collision Avoidance). Siden nodene ikke kan detektere kollisjoner må de prøve å unngå dem fullstendig. Ved bruk av CSMA/CA fortsetter ikke noden å lytte hvis den oppfatter at

kanalen er opptatt, men trekker seg heller tilbake i en tidsperiode av tilfeldig lengde. I det ventetiden har gått ut vil den på nytt sjekke om kanalen er ledig, og eventuelt beregne en ny tilfeldig venteperiode om nødvendig. Denne prosessen gjentas til kanalen er ledig og overføringen gjennomføres [17].

3.1.3 TDMA

I likhet med CSMA er TDMA (Time Division Multiple Access) en protokoll for å styre tilgangen til et delt medium for nodene i nettverket i den hensikt å unngå interferens og kollisjoner. I tillegg kan bruken av TDMA-protokollen sørge for et deterministisk nettverk, der nettverksnodene på forhånd vet når de kan sende og når de kan forvente å motta data. Dette kan ha sine fordeler i et trådløst sensornettverk, der det ofte er ønskelig å holde radioen mest mulig avslått for å spare batteribruk.

Hovedprinsippet i TDMA er å dele kanaltilgangen inn i tidslommer av fast lengde, gruppere disse tidslommene i rammer med fast antall lommer, og å sørge for at bare en node kan sende på det delte mediet innenfor hver tidslomme. En veldig enkel implementering av en TDMA-protokoll i et nettverk med fire noder er illustrert i figuren nedenfor, der hver node har fått tildelt hver sin dedikerte tidslomme som kan brukes til sending.



Figur 3-2 Enkel bruk av TDMA

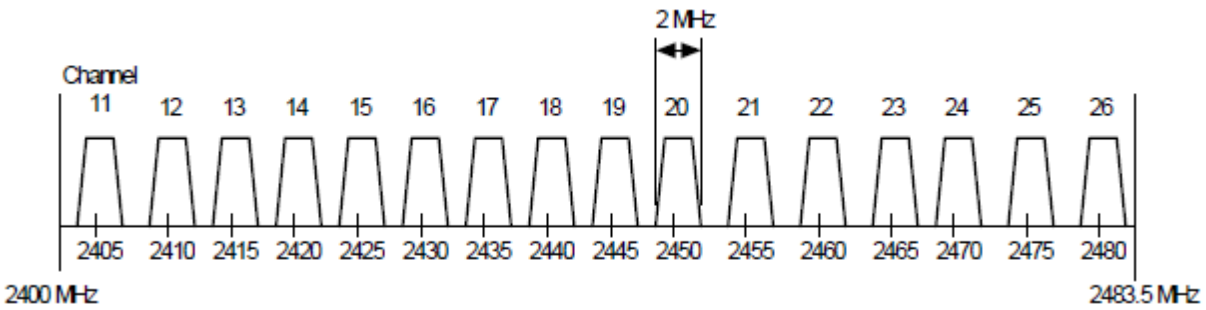
En noe mer avansert TDMA-protokoll er den som brukes i WirelessHART [20]. Vi skal se på den i mer detalj senere, men det kan være verdt å merke seg at WirelessHART opererer med både dedikerte sendetidslommer og dedikerte mottakstidslommer. Siden nodene i nettverket på forhånd dermed vet både når det er aktuelt å ta i mot data og når de kan sende data, kan de

slå radioene helt av i mellomtiden. Dette har åpenbare fordeler med tanke på energiforbruk. Ulempen med å implementere TDMA på denne måten er at tidssynkronisering på nettverket blir kritisk, siden både sender og mottaker må være samkjørt om når de ulike tidslommene starter for å kunne slå radioen av og på til rett tid. Dette forvanskes ytterligere av at nettverksnodene hele tiden må kompensere for at de ulike klokkene driver fra hverandre i tid.

3.1.4 Kanaler

En kanal i forbindelse med trådløse nettverk er i praksis et frekvensområde. Det tilgjengelige frekvensbåndet deles inn i mindre, adskilte kanaler. Dette kan, som i protokollen FDMA (Frequency-division medium access), utnyttes til å sende flere dataforsendelser parallelt [21], eller til å la ulike nettverk i samme frekvensområde operere på ulike kanaler for å unngå interferens. Kanalene danner også grunnlaget for den neste protokollen vi skal se på, nemlig kanalhopping.

Antall kanaler trafikken kan fordeles på avhenger av tilgjengelig båndbredde og bredden på hver kanal. IEEE 802.15.4-standarden har 16 kanaler på 2,4 GHz-båndet (kanal 11 til 26) der hver kanal er 2 Mhz bred og hvor det skiller 5 MHz mellom midtpunktene til to nabokanaler. (For eksempel er kanal nummer 11 å finne på 2405 MHz og kanal nummer 12 er 5 MHz «unna» på 2410 MHz) [22]. Bredden på kanalen (2 MHz) er større enn nødvendig for dataoverføring, noe som skyldes transmisjonsteknikken som brukes. DSSS (Direct Sequence Spread Spectrum) «smører» signalet over et bredere frekvensbånd, for så å addere det sammen igjen på mottakersiden. På denne måten blir signalet mindre utsatt for støy og interferens på enkeltfrekvenser.



Figur 3-3 Kanalene på 2,4 Ghz-båndet I IEEE 802.15.4 [18]

3.1.5 Kanalhopping

Kanalhopping er en teknikk for hyppig endring av sende-/mottaksfrekvens. Hensikten med kanalhopping er primært å bekjempe interferens og signaldegradering ved multiple stier (multipath fading) [17]. Signaldegradering ved multiple stier er et fenomen som oppstår når radiosignaler reflekteres av objekter, som igjen fører til at de samme signalene blir mottatt flere ganger fra forskjellige stier og dermed skaper interferens. Ved kanalhopping motvirker de hyppige frekvensskiftene denne effekten. Kanalhopping vil dessuten gi økt kapasitet i nettverket kontra et nettverk som kun forholder seg til én kanal, siden flere enheter kan kommunisere samtidig på ulike frekvenser.

Problemer knyttet til interferens kan for eksempel være annen trådløs dataoverføring som opererer på samme frekvens, noe som lett fører til store pakketap og dermed dårlig nettverksytelse. Siden kanalhoppingsprotokoller hele tiden endrer frekvens vil dette problemet bli betraktelig mindre. I tillegg gir kanalhopping også en økt sikkerhet, fordi hoppingen mellom kanalene foregår i en pseudo-tilfeldig rekkefølge som er ukjent for en eventuell inntrenger. Avlytting av trafikken blir dermed vanskelig.

I enkelte protokoller benyttes en transmisjonsteknikk kalt FHSS (Frequenzy Hopping Spread Spectrum) som er beslektet med kanalhopping, men denne virker i selve radioen og hopper mellom mange kanaler med smalere frekvensbånd [17]. Kanalhopping i den formen vi møter den er en protokoll på MAC-laget, og kanalene det veksles mellom er, sammenlignet med FHSS-protokollen, relativt få og brede.

3.2 IEEE 802.15.4

Den første IEEE 802.15.4-standarden, «Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)» [23] ble gitt ut i 2003, og tittelen levner liten tvil om hva den dekker. Prosjektgruppen sier selv at formålet var å definere det fysiske laget og datalinklaget for nettverk med høye krav til lavt strømforbruk, med enheter som typisk opererte innenfor det personlige arbeidsområdet (10 m) og som kunne være faste, portable eller i bevegelse. I tillegg måtte det kunne sameksistere med andre trådløse enheter og teknologier som for eksempel tradisjonell WLAN. Mer spesifikt var hensikten å tilby en standard for ultralav kompleksitet, kostnad og strømforbruk for trådløs kommunikasjon mellom billige enheter. Datahastigheten på maksimum 250 kb/s skulle være nok til eksempelvis interaktive leker, og skalerbar ned til 20 kb/s eller lavere for bruk til blant annet trådløse sensornettverk.

I 2006 kom en ny versjon av 802.15.4-standarden, som for det meste er mindre justeringer av den opprinnelige. Blant annet er det gjort noen endringer på det fysiske laget for å slippe til flere frekvensbånd og for å forbedre utnyttelsen av de eksisterende. I tillegg er det ryddet opp i en del tvetydigheter, noe unødvendig kompleksitet er fjernet, og det er innført forbedringer i sikkerhetsnøkkelbruken. Videre i teksten benytter vi IEEE 802.15.4 som betegnelse der de nye justeringene ikke er av betydning. Om det derimot er relevant å spesifisere hvilken versjon av standarden som gjelder blir dette spesifisert som IEEE 802.15.4-2003 og IEEE 802.15.4-2006.

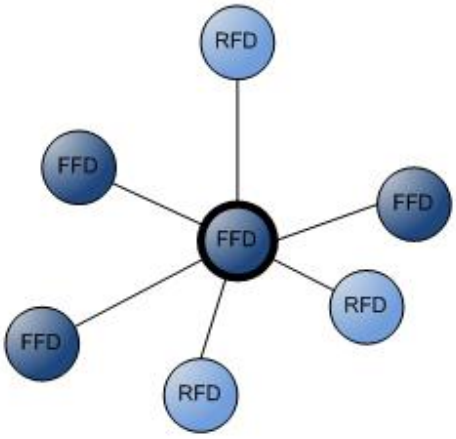
3.2.1 Komponenter og topologier

Komponenter

Et IEEE 802.15.4-nettverk består av et antall IEEE 802.15.4-enheter. Det skal alltid være nøyaktig én nettverkskoordinator, kalt PAN-koordinator, i hvert nettverk, og et nettverk må i tillegg ha minimum én nettverksenhet. I tillegg til dette definerer standarden to typer av nettverksenheter; Full Function Device (FFD) og Reduced Function Device (RFD). Den fullt funksjonelle enheten, altså FFD-en, inneholder hele settet av MAC-tjenester, og dette gjør at den i tillegg til å være en vanlig nettverksnode også kan fungere som nettverkskoordinator. En node med redusert funksjonalitet, en RFD, er designet for å være ekstremt enkel med tanke på å holde ressursforbruket nede, og kan bare fungere som nettverksenhet.

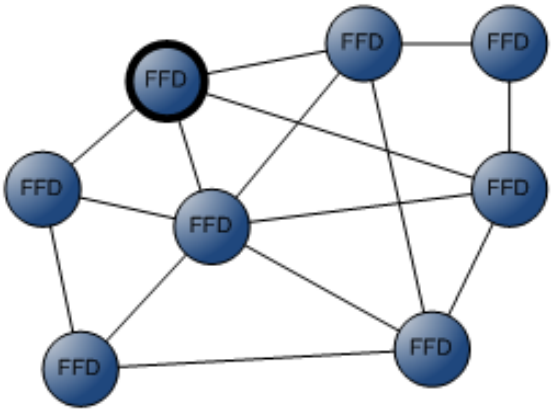
Topologier

Det finnes to grunnleggende topologier for IEEE 802.15.4-nettverk. Den enkleste kalles stjernetopologi, og består av en FFD som fungerer som PAN-koordinator, omgitt av RFD-er eller andre FFD-er. I denne topologien er koordinatoren den eneste som har en direkte link til mer enn én annen enhet, og all trafikk går derfor gjennom den. Dette er illustrert i Figur 3-4, med PAN-koordinatoren markert med tykk ramme.

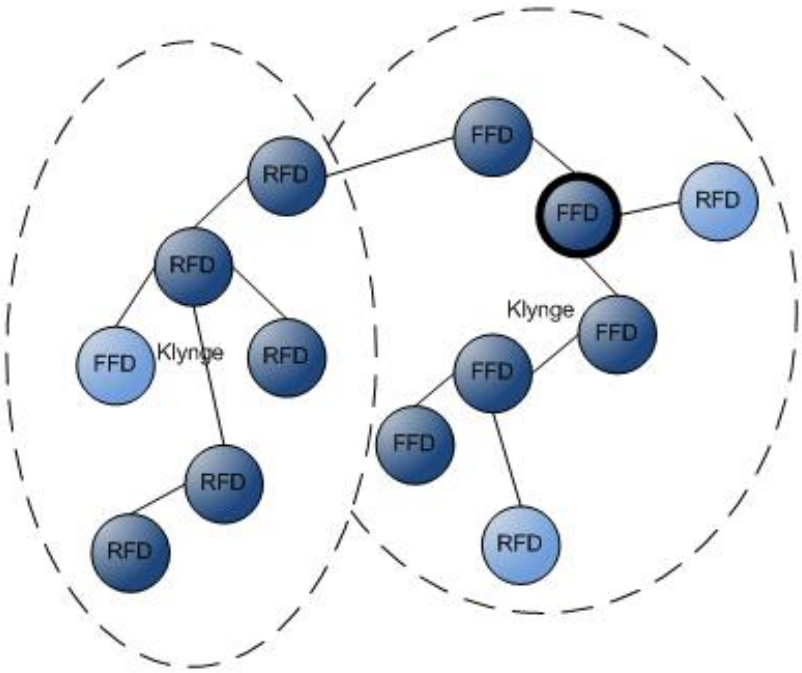


Figur 3-4 Stjernetopologi

Den andre topologien, «peer-to-peer», tillater at nodene kommuniserer direkte med hverandre uten å gå via en PAN-koordinator. Det må fortsatt være en PAN-koordinator et eller annet sted i nettverket, men alle nodene har muligheten til å opprette flere tilkoblinger til andre enheter. Dette danner grunnlaget for ulike varianter av peer-to-peer-topologier, eksempelvis maske- og klyngetre-topologi som vist i Figur 3-5 og Figur 3-6. Nodene i slike nettverk må være FFD-er, siden de må være i stand til å rute og videresende meldinger. Unntaket er eventuelle bladnoder, som ikke vil ha behov for disse videresendingsegenskapene.



Figur 3-5 Masketopologi



Figur 3-6 Klyngetre-topologi

3.2.2 Det fysiske laget (PHY)

Som vi allerede har vært inne på dekker IEEE 802.15.4-standarden det fysiske laget og MAC-laget for protokollstakken til et trådløst sensornettverk. Vi vil i dette underkapittelet forsøke å gi en kortfattet og overordnet beskrivelse av de viktigste tjenesteprimitivene og mekanismene på det fysiske laget, før vi i neste underkapittel gjør en tilsvarende gjennomgang av MAC-laget.

Det fysiske laget er, som vist i Tabell 3-1, det nederste laget i ISO/OSI-modellen. IEEE 802.15.4 PHY har ansvar for følgende oppgaver:

- Aktivere og deaktivere radiosender/-mottaker
- Energidetektering innen den aktuelle kanalen
- Linkkvalitetsindikasjon for mottatte pakker
- Vurdere om kanalen er ledig
- Velge kanalfrekvens
- Datasending og –mottak

Av årsaker relatert til lokale reguleringer i deler av verden, spesifiserer standarden tre ulike frekvensområder som kan benyttes. Dette er 868 MHz-båndet, 915 MHz-båndet og 2,4 GHz-

båndet, inndelt i to ulike fysiske sublag slik Tabell 3-2 og Tabell 3-3 viser. Siden 2,4 GHz-båndet er et av de såkalte ISM-båndene (industrial, scientific and medical) og er åpent tilgjengelig i de aller fleste land i verden, har de fleste som benytter standarden valgt å bruke dette. Produkter kan da selges over stort sett hele verden, i tillegg til å passere landegrenser uten problemer, og dermed blir både markedet større og produksjonskostnadene lavere. De ulike bitratene i Tabell 3-3 skyldes at IEEE 802.15.4-2006 innførte muligheten til å velge modulasjonsteknikker på de laveste frekvensbåndene som tillater høyere bitrater enn tidligere, dog på bekostning av den lavere kompleksiteten i IEEE 802.15.4-2003.

PHY sublag	Frekvensbånd (MHz)	Bitrate	Antall kanaler
868/915 MHz	868-868.6 MHz	20 kb/s	1
	902-928 MHz	40 kb/s	10
2450 MHz	2400-2483.5 MHz	250 kb/s	16

Tabell 3-2 IEEE 802.15.4-2003 fysiske sublag

PHY sublag	Frekvensbånd (MHz)	Bitrate	Antall kanaler
868/915 MHz	868-868.6 MHz	20, 100, 250 kb/s	1
	902-928 MHz	40, 250 kb/s	30
2450 MHz	2400-2483.5 MHz	250 kb/s	16

Tabell 3-3 IEEE 802.15.4-2006 fysiske sublag

Det kreves i spesifikasjonen at radioene minst må kunne overføre med en styrke på -3 dBm, og være i stand til korrekt å dekode et signal med en styrke på -85 dBm eller mindre i 2,4 GHz-båndet. Dette gir en teoretisk rekkevidde på drøye 200 meter for 2,4 GHz-radioer, men i praksis vil rekkevidden alltid være betraktelig lavere [8].

Tjenester

I IEEE 802.15.4 fungerer det fysiske laget som et grensesnitt mellom den fysiske radiokanalen og MAC-laget via to tjenester; PHY datatjeneste og PHY administrasjonstjeneste. Datatjenesten står for overføringen av datapakker, mens administrasjonstjenesten tilbyr kanalinformasjon til overliggende lag, samt muligheter for endringer av konfigurasjonsverdier i det fysiske lagets informasjonsbase. Sistnevnte er en

datastruktur kalt PIB (Personal Area Network Information Base) som inneholder relevant konfigurasjonsinformasjon for det fysiske laget. Eksempelvis hvilken kanal radioen skal benytte, eller hvilken overføringseffekt som skal benyttes.
IEEE 802.15.4 benytter seg av tjenesteprimitiver slik det er forklart i underkapittel 3.1.1. Altså er alle tjenestene både i PHY og MAC bygget rundt en kombinasjon av de fire tjenesteprimitivene *request*, *indication*, *response* og *confirm*. I PHY datatjeneste benyttes tre av dem, og de er kort oppsummert spesifisert som dette:

- *PD-DATA.request.* Krever parametrene lengde og dataenhet (pakke). Genereres hos det lokale MAC-laget og sendes til PHY for å be om transmisjon av en datalinklagspakke. Forutsatt at radioen står i sendemodus (TX_ON) vil PHY legge på PHY-hode og overføre. Når dette er fullført sendes en PD-DATA.confirm til overliggende lag med SUCCESS som status. Skulle radioen stå i mottaksmodus (RX_ON), være avslått (TRX_OFF) eller opptatt med å sende en annen pakke (BUSY_TX) vil ikke forespørselen kunne behandles, og en PD-DATA.confirm sendes opp med henholdsvis RX_ON, TRX_OFF eller BUSY_TX som status.
- *PD-DATA.confirm.* Krever status som parameter. Genereres fra PHY som svar på en PD-DATA.request. Status kan være SUCCESS, RX_ON, TRX_OFF eller BUSY_TX.
- *PD-DATA.indicate.* Krever dataenhet, datalengde og linkkvalitet som parametere. Genereres fra PHY og sendes til MAC ved mottak av en pakke. Gir MAC beskjed om at en ny pakke har kommet.

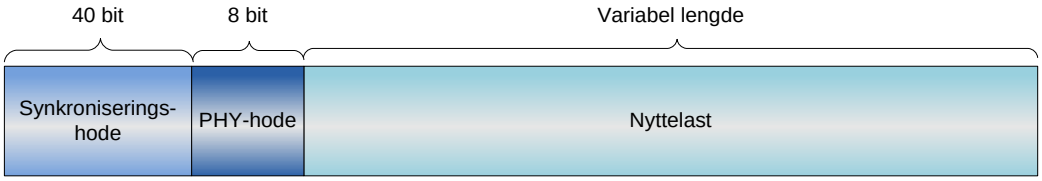
PHY administrasjonstjeneste har fem undertjenester, hvor hver av dem er spesifisert gjennom primitivene *request* og *confirm*. Disse fem tjenestene kan kort oppsummeres slik:

- *PLME_GET.* Request-primitivet ber PHY om verdien på et PIB-attributt. Eneste parameter er PIB-attributt-ID. PHY svarer med en confirm med samme ID som parameter, samt selve verdien på dette attributtet. I tillegg sendes også en status med som parameter, som settes til SUCCESS i normale tilfeller og til UNSUPPORTED_ATTRIBUTE hvis attributt-ID ikke finnes.
- *PLME_SET.* Tilsvarende som GET, men setter en verdi i stedet for å lese den ut. Request tar derfor et attributt og en attributt-ID som parametere, mens confirm tar status og attributt-ID. Status er samme som i PLME_GET.

- *PLME_SET_TRX_STATE.* Request tar en status som parameter og ber radioen endre sin status til dette. Mulige verdier er RX_ON, TRX_OFF, FORCE_TRX_OFF og TX_ON. Confirm vil svare med SUCCESS hvis radioen endret status, og med RX_ON, TRX_OFF eller TX_ON hvis radioen allerede befant seg i den forespurte statusen.
- *PLME_CCA.* Request-primitivet ber PHY utføre CCA (Clear Channel Assesment sjekker at kanalen ikke er i bruk). Svarer med en confirm med status TRX_OFF, BUSY eller IDLE. Brukes av CSMA/CA-algoritmen.
- *PLME_ED.* Request-primitivet ber PHY utføre en ED-måling (ED – deteksjon av energinivået på aktuell kanal). Svarer med en confirm med status SUCCESS, TRX_OFF eller TX_ON, samt en verdi som forteller ED-nivået om status er SUCCESS.

PHY-pakkens struktur

Pakken som sendes ut på det fysiske mediet (PHY Protocol Data Unit – PPDU) er relativt enkel i sin konstruksjon (se Figur 3-7). Den består av tre komponenter: Et synkroniseringshode, et PHY-hode som angir pakkens lengde, og til slutt selve nyttelasten av variabel lengde. Nyttelasten vil i de fleste tilfeller være en pakke fra MAC-laget.



Figur 3-7 PPDU-struktur

3.2.3 Medium Access Control-laget (MAC)

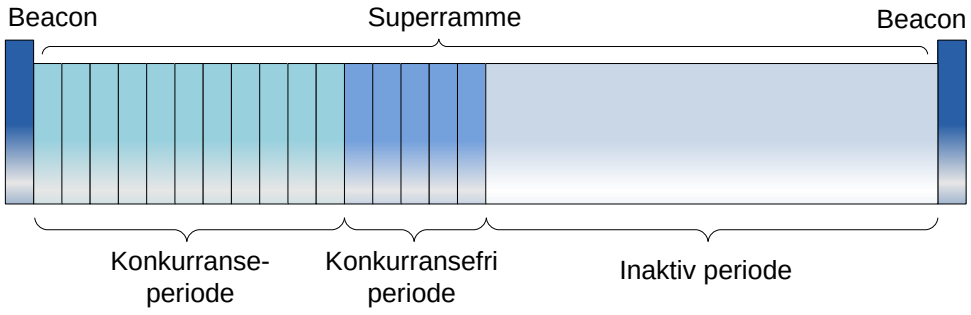
Som vi tidligere har vært inne på er datalinklaget i IEEE 802-modellen delt inn i et MAC-lag og et LLC-lag (logisk linkkontroll). Hovedoppgaven til LLC-laget er å tilby multipleksing og flytkontroll, samt fungere som et grensesnitt mellom nettverkslaget og MAC-laget. IEEE 802.2-standarden [16] (som definerer LLC) ble imidlertid skrevet før WPAN så dagens lys, og siden man i trådløse sensornettverk gjerne er ute etter enkelhet, lavt strømforbruk og rask utvikling av enheter, er mye av funksjonaliteten i LLC-laget forenklet og trukket inn i IEEE 802.15.4 MAC-laget. Dette gjør det mulig å sløyfe LLC-sublaget og legge nettverkslaget

direkte på MAC-laget. Vi vil derfor se bort fra IEEE 802.2-standarden i denne sammenhengen.

MAC-laget tilbyr tilgangskontroll til en delt kanal og sørger for pålitelig dataoverføring. Tilgangskontrollen utføres med tanke på optimal utnyttelse av nettverket, noe som krever en viss etikette av de ulike enhetene, siden trådløs nettverkstrafikk er utsatt for støy og annen trafikk på samme frekvens. I IEEE 802.15.4 løses dette med den tidligere omtalte kollisjonsunngåelsesalgoritmen CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance).

I tillegg legger standarden til rette for en superramme-struktur med såkalte «beacons». En beacon er en melding fra PAN-koordinatoren til alle enhetene som blant annet inneholder synkroniseringsdata og nettverks-ID, mens superrammen er tidsperioden mellom to slike meldinger. En superramme består i IEEE 802.15.4 av 16 tidslommer åpne for kommunikasjon mellom nodene, etterfulgt av en inaktiv periode hvor nodene har muligheten til å slå av radiosenderen og spare strøm. Den inaktive perioden kan være av valgfri lengde, eller også sløyfes helt.

De 16 tidslommene i den aktive perioden er i utgangspunktet konkurransebasert, altså at nodene «slåss» om å få bruke dem ved hjelp av CSMA/CA-prosedyren, men PAN-koordinatoren kan fordele inntil syv av dem som såkalte garanterte tidslommer til enheter som har spesielle behov. Disse vil i så fall alltid komme sist i den aktive perioden. Et eksempel på en superramme med fem garanterte tidslommer og en inaktiv periode er vist i Figur 3-8 under. Bruken av superrammer er valgfri i IEEE 802.15.4, men valget gjelder da for samtlige noder på nettverket.



Figur 3-8 Superramme i IEEE 802.15.4

For øvrig er det verdt å merke seg at PAN-koordinatoren må vente på en dataforespørsel fra nettverksnodene før den kan sende vanlige datarammer tilbake, siden nodene ellers kan være i

en tilstand der de ikke er i stand til å ta i mot data. Dette gjelder ikke andre veien, siden PAN-kooridinatoren alltid er på.

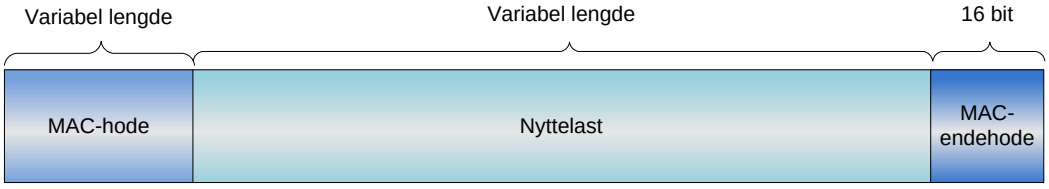
Tjenester

IEEE 802.15.4 MAC tilbyr to typer tjenester til høyereliggende lag: MAC datatjeneste og MAC administrasjonstjeneste. MAC datatjeneste tilbyr tjenesteprimitiver for dataoverføring, og er i stand til å håndtere ulike adresseringsteknikker for å gjøre det enkelt å sette opp forskjellige nettverkstopologier. MAC administrasjonstjeneste tilbyr muligheter til å håndtere kommunikasjonsinnstillinger, radioen og nettverksfunksjonalitet. Herunder assosiasjon og disassosiasjon av enhet til PAN-koordinator, håndtering av garanterte tidslommer i superramme-nettverk, håndtering av enheter som mister kontakt med koordinator, synkronisering med og uten beacons, samt håndtering av beacons.

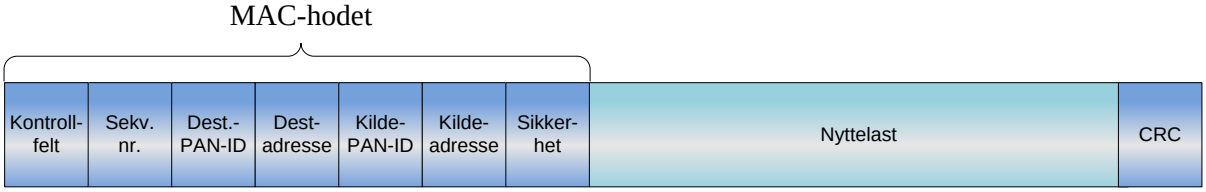
MAC-rammens struktur

MAC-rammen består, som vist i Figur 3-9, av tre hoveddeler: et hode, nyttelasten, og et endehode. Vi skal ikke her gå i særlig detalj på oppbyggingen av hodene, men kort fortalt innholder MAC-hodet et kontrollfelt for å spesifisere blant annet pakketype og adresselengde, et felt med sekvensnummer, til sammen fire felt for avsender- og mottakeradresser for PAN og node, samt et felt som forteller om bruk av IEEE 802.15.4s sikkerhetsmekanismer i kommunikasjonen (Figur 3-10). Endehodet inneholder feilsjekkingsdata (CRC).

Den ferdige MAC-rammen sendes ned til PHY og pakkes der inn med synkroniseringshodet og PHY-hodet til å bli en fullverdig PHY-pakke.



Figur 3-9 MAC-rammens struktur



Figur 3-10 MAC-hodet

3.3 ZigBee

ZigBee er en standard for LR-WPAN (Low-Rate Wireless Personal Area Network) rettet mot hjemme-automatisering, overvåking og kontroll. Den har eksistert siden 2004, og er basert på IEEE 802.15.4-standarden fra året før. ZigBee er sannsynligvis den mest kjente og utbredte fullstendige protokollstakk for denne type nettverk, og det finnes en lang rekke leverandører av utstyr klargjort for ZigBee. Bak ZigBee står et konsortium av en rekke produsenter som sammen danner «The ZigBee Alliance» og som har ansvar for utgivelse og vedlikehold av standarden.

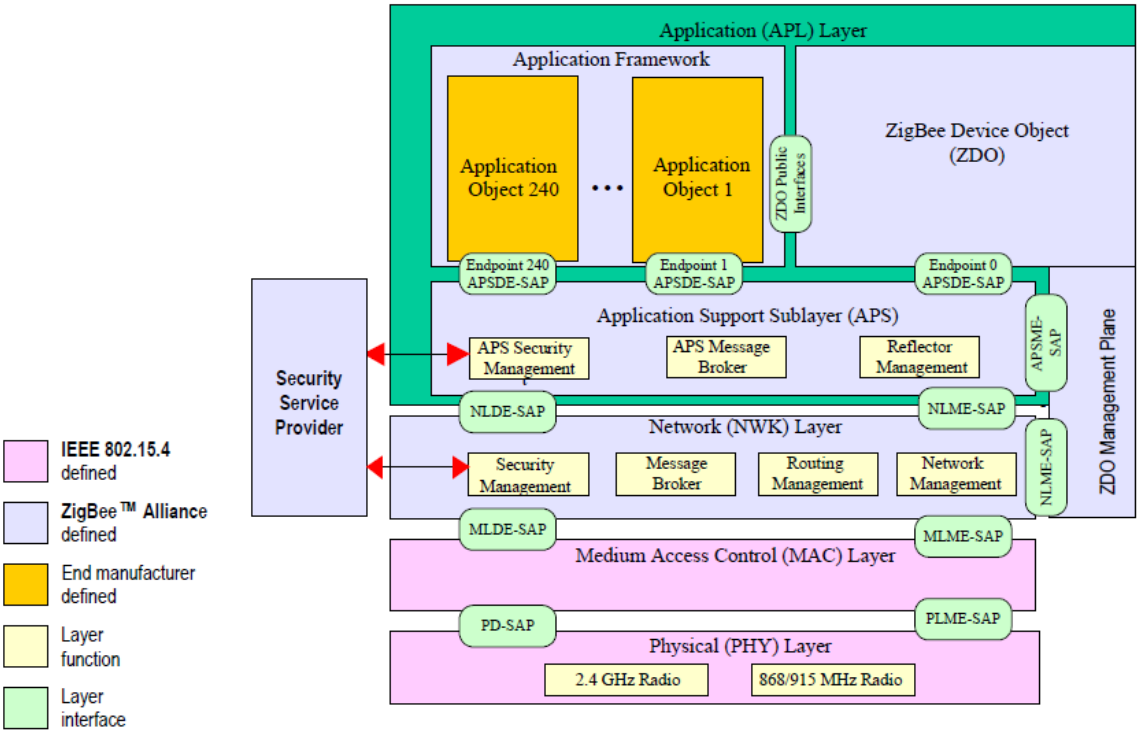
ZigBee var en av de tidligste spesifikasjonene som tok utgangspunkt i IEEE 802.15.4. I følge The ZigBee Alliance selv er standarden beregnet på forbrukerelektronikk, hjem- og bygningsautomatisering, industriell kontroll, PC-utstyr, medisinske sensorapplikasjoner, leker og spill [24]. Til bruk i industrielle sensornettverk har imidlertid ZigBee blitt vurdert som ikke robust nok [25], og i 2007 ble ZigBee-spesifikasjonen derfor utvidet til ZigBee PRO som blant annet inneholdt utvidede mekanismer for sikkerhet og robusthet. PRO-profilen nevner «sensing and control in commercial, industrial and institutional environments» som et hovedbruksområde, sammen med bygningsautomatisering [26].

Som vist i Tabell 3-1 på side 13 benytter ZigBee hele IEEE 802.15.4-2003 som det fysiske laget og datalinklaget, og har sitt eget nettverkslag og applikasjonslag på toppen av dette [24]. I likhet med IEEE 802.15.4 støtter ZigBee stjerne-, tre-, og masketopologier. Alle nettverk har en ZigBee-koordinator, og i et stjernenettverk går all trafikk gjennom denne. I tre- og maskenettverk er koordinatoren ansvarlig for å starte nettverket og for valg av enkelte nettverksparametere, men i tillegg brukes det her ZigBee-rutere for å utvide nettverket slik at ikke alle må gå via koordinatoren. En ZigBee-koordinator er i praksis det samme som en IEEE 802.15.4 PAN-koordinator, og en ZigBee-ruter er en fullt funksjonell IEEE 802.15.4-enhet (FFD) som derfor har rutingkapabilitet. Endenodene i ZigBee kan være enten noder med redusert funksjonalitet (RFD) eller fullt funksjonelle noder (FFD).

Siden alt dette er MAC-spesifikt er det så langt bare navnene på komponentene som skiller ZigBee fra IEEE 802.15.4. IEEE 802.15.4 kan imidlertid ikke alene tilby nettverk med flere hopp og maskenettverk. Standarden muliggjør topologiene beskrevet i 3.2.1, men de trenger et nettverkslag til å sikre korrekt bruk av dem, samt håndtere konfigurasjon av nye enheter og etablering av nye nettverk. ZigBees nettverkslag gjør nettopp dette, i tillegg til å fungere som et grensesnitt mellom applikasjonslaget og datalinklaget. ZigBees applikasjonslag er grensesnittet opp til brukeren, og består av tre hoveddeler:

- *Applikasjonsrammeverket.* Et miljø der applikasjoner kan kjøre.
- *ZigBee enhetsobjekt.* Definerer enhetens rolle.
- *Applikasjonsstøttesublag.* Grensesnitt mellom nettverkslaget og applikasjonslaget.

Innenfor applikasjonsrammeverket er det åpent for produsenter å definere inntil 240 applikasjonsobjekter som håndteres på enhetene. På toppen av dette kan det defineres applikasjonsprofiler som sikrer interoperabilitet mellom sertifiserte enheter som deler samme profil. Figur 3-11 er hentet fra ZigBee-standarden og viser protokollens arkitektur [24].



Figur 3-11 ZigBees protokollarkitektur [24]

3.4 WirelessHART

Vi vil gå grundig gjennom WirelessHART-standarden i et eget kapittel senere i oppgaven (kapittel 0), og nøyer oss derfor med en beskrivelse av noen generelle trekk i denne sammenheng. Som nevnt tidligere ble standarden utgitt i 2007, og bak sto HART Communication Foundation (HCF), en non-profit-organisasjon hovedsakelig bestående av utstyrsprodusenter og brukerselskaper. HCF forvalter også HART-protokollen, som er en eldre og mye utbredt trådbasert protokoll for industrielle sensornettverk. WirelessHART

fungerer som den trådløse adapteringen av HART, og muliggjør kombinasjonen av trådløse og trådbaserte enheter i samme nettverk. Protokollen bærer selvfølgelig preg av dette, og det finnes ingen støtte for andre applikasjonsprotokoller enn HARTs eget, slik ISA100.11a har (se kapittel 3.5).

Det fysiske laget til WirelessHART er basert på IEEE 802.15.4-2006 2,4 GHz. Ut over dette definerer standarden et eget datalinklag og nettverkslag, mens den opprinnelige HART-protokollens transportlag og applikasjonslag fungerer på toppen av dette.

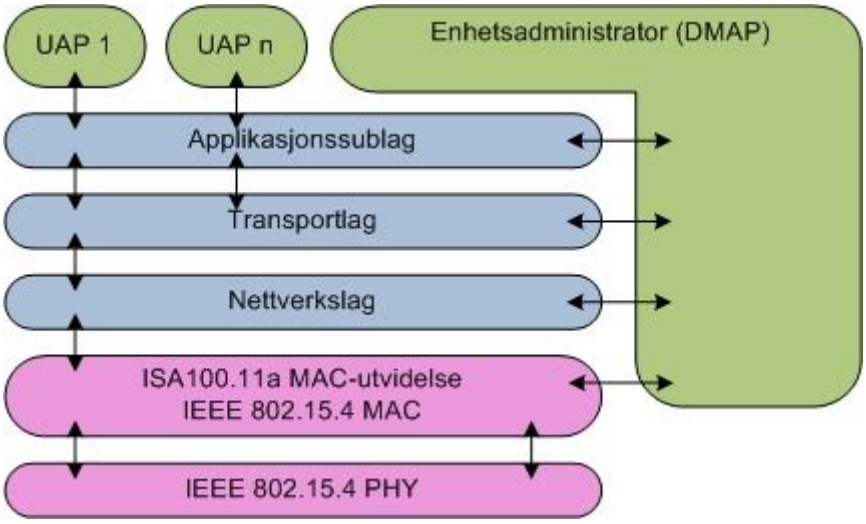
En av WirelessHARTs viktigste egenskaper er kombinasjonen av TDMA og kanalhopping på datalinklaget. Med TDMA i den formen det benyttes i WirelessHART, der både sende- og lyttetidspunkter er definert for alle nodene, blir nettverket svært deterministisk og tillater at noder kan sove mesteparten av tiden. Kanalhopping bidrar til bedre sikkerhet med tanke på avlytting, samt bedre robusthet når nettverket skal eksistere i samme miljø som andre teknologier på samme frekvensbånd [27]. I tillegg åpner kanalhopping for at flere datautvekslinger kan foregå på samme tid, forutsatt at ulike kanaler brukes. Det er verdt å merke seg at kanalhopping også vil kunne frembringe en flaskehals nær noder som fungerer som mottaker for mange stier (for eksempel i nærheten av nettverksadministratoren), siden disse nodene kun kan lytte på en kanal av gangen.

3.5 ISA100.11a

ISA100.11a, også kjent som SP 100, er den ferskeste av de omtalte standardene. Den ble ratifisert av ISA (The International Society of Automation) så sent som i september 2009 [28], og kom som en følge av et ønske om konvergens mellom flere trådløse industristandarder som var under utvikling. Protokollen har mange likhetstrekk med WirelessHART både i bruksområder og tekniske løsninger, men der WirelessHART er ment som en standard for trådløs overføring av HART-kommandoer, er ISA100.11a ment å være en felles standard for trådløs overføring av flere ulike industrielle applikasjonsprotokoller. HART, Foundation Fieldbus, og PROFIBUS er blant de protokollene som støttes, i tillegg til ISA100-applikasjonsprotokollen. Sammenlignet med WirelessHART er standarden i større grad også rettet mot å være et kontrollnettverk, og ikke bare primært et sensornettverk for overvåking.

Rent teknisk er protokollen spesifisert med fokus på sameksistens med andre trådløse teknologier. Også ISA100.11a benytter 2,4 GHz-båndet som beskrevet i IEEE 802.15.4-2006 som det fysiske laget. I tillegg benyttes kanalhopping og svartelisting av kanaler på

datalinklaget for ytterligere å øke robustheten mot interferens. Arkitekturen består, i tillegg til IEEE 802.15.4-2006 PHY, av et datalinklag, et nettverkslag, et transportlag og et applikasjonslag. Disse fire lagene, som er ISA100.11a-spesifikke, tilbyr en direkte tilgang til interne administrasjonstjenester. Administrasjonstjenestene går under betegnelsen DMAP (Device-management application process). DMAP danner grunnlaget for en applikasjon som kan utføre systemadministrering og overvåking av kommunikasjon på enheten. Datalinklaget er todelt, med IEEE 802.15.4-2006 MAC som grunnlag, og en egen ISA100.11a-utvidelse på toppen av dette. På toppen av applikasjonslaget kan produsenter legge sine egne brukerapplikasjonsprosesser (UAP – User application process), ikke ulikt applikasjonsobjektene i ZigBee. Figur 3-12 er hentet fra en studie på ISA100.11a og sikkerhetsmekanismer og viser ISA100.11a-arkitekturen [29].



Figur 3-12 Forenklet ISA100.11a protokollarkitektur [29]

ISA100.11a definerer to hovedklasser av enheter: feltenheter og kjerneenheter (field devices og backbone devices). En feltenhet kan være både en FFD og en RFD som definert i IEEE 802.15.4, mens kjerneenhetene er FFDer med fast strømtilførsel. En spesiell kjerneenhet fungerer som systemadministrator og som klokkekilde i forbindelse med tidssynkronisering.

3.6 Protokollene sammenlignet

Det er vanskelig å trekke noen konklusjoner om hvilken av de tre nettverksprotokollene ZigBee, WirelessHART og ISA100.11a som er best. Alle har sine fordeler og ulemper, og alle

har sterke støttespillere bak seg. Tidligere studier [25, 30] har imidlertid vist at ZigBee ikke er like godt egnet fra et industrielt perspektiv. Som vi så i kapittel 2.3 er koeksistens og strømforbruk blant de viktigste kravene, og det er her ZigBees mangler viser seg. Siden ZigBee implementerer hele IEEE 802.15.4-protokollen forsvinner muligheten til reell kanalhopping. Hele nettverket deler den samme statiske kanalen, noe som gjør det utsatt for jamming, samt for interferens fra WLAN og andre systemer på 2,4 GHz-båndet. Grunnet de metallrike omgivelsene i industriell sammenheng kan også signalsvekking ved multiple stier (multipath fading) være et potensielt problem. I tillegg vil strømforbruket kunne bli høyt i nettverk med høy nodetetthet, siden bruken av CSMA/CA i slike tilfeller kan føre til at radiomottakeren må benyttes mye for å identifisere ledige tidsrom for overføring av data.

Denne kritikken ble delvis imøtegått med ZigBee PRO [26] i 2007, en utvidelse av den opprinnelige protokollen spesifikt for trådløse sensornettverk til industribruk. ZigBee PRO innførte blant annet bruk av såkalt «frekvenssmidighet», som sørger for at nettverket vil skifte kanal hvis det registrerer interferens eller støy, og PRO medførte dermed en økning i robusthet. PRO legger også til rette for bedre energiutnyttelse ved en mer utstrakt bruk av sovemekanismene enn tidligere. Likevel er det fortsatt slik at noder som skal ha rutingegenskaper (FFD-er) alltid må være på, og derfor må ha konstant strøm eller hyppige batteriskifter. Alle ZigBee PRO-nettverk av en viss størrelse vil benytte FFD-er, og i mange industrielle sammenhenger er det lite gunstig å måtte ha disse koblet til strømnettet eller være avhengig av å skifte batterier ofte. For sensornettverk hvor batterilevetid er et viktig krav, slik det blant annet er i olje- og gassindustrien (se kapittel 2.3), er det derfor mer aktuelt å vurdere WirelessHART eller ISA100.11a.

ISA100.11a er som tidligere nevnt en veldig fersk standard, og det er derfor vanskelig å si noe konkret om hvor bra den fungerer i praksis siden lite arbeid er gjort på dette. Det er imidlertid ingen tvil om at standarden virker lovende, og den ser ut til å ta for seg de viktigste kravene fra et industrisynspunkt. WirelessHART har vært på markedet noe lenger, og har ved flere gjennomganger kommet positivt ut med tanke på industrielle krav [25, 27, 31]. Fra et overordnet perspektiv er de tekniske forskjellene ikke enorme. Begge er basert på IEEE 802.15.4-2006, begge benytter kanalhopping for god robusthet, begge har omfattende sikkerhetsmekanismer, og begge skalerer godt. ISA100.11a har en åpenbar fordel i at den støtter flere applikasjonslagsprotokoller (mens WirelessHART kun støtter HART), men samtidig stiller dette større krav til produsenter som får en langt mer omfattende implementeringsjobb.

I bransjemiljøer har WirelessHART og ISA100.11a ofte blitt sett på som direkte konkurrenter i en kamp om å bli den globale standarden for industriell trådløs automatisering [32], og selv om WirelessHART var først ute med en standard og dermed må sies å ligge et hestehode foran, er ISA100.11a en sterk utfordrer med sin mer universelle applikasjonsstøtte. I 2007 inngikk imidlertid HCF og ISA et samarbeid med sikte på å utvikle en felles trådløs standard for prosessmåling og kontrollapplikasjoner [33]. ISA satte sågar ned en arbeidsgruppe, ISA100.12, for å se på mulighetene for konvergens mellom ISA100.11a og WirelessHART med sikte på å gjøre WirelessHART til et fullverdig subsett av ISA100.11a. I januar 2010 hadde ControlGlobal.com en lengre artikkel [34] om blant annet ISA100.11a og forholdet til WirelessHART der det kom frem at dette arbeidet virket å være vanskelig, samtidig som ISA100.11as vanskeligheter med å bli tatt opp som en IEC-standard (International Electrotechnical Commission) ble beskrevet. Dette dreide seg først og fremst om formaliteter i forbindelse med forsøket på å bli tatt opp som en ANSI-standard (American National Standards Institute), antatt å være en forutsetning for å oppnå IEC-standardisering, men artikkelen antydet likevel at WirelessHART kom styrket ut av denne perioden rundt årsskiftet 2009/2010. Det ble blant annet pekt på at WirelessHART på dette tidspunktet allerede var godt inne i sin IEC-standardiseringsprosess (WirelessHART ble standardisert to måneder senere, i mars 2010, etter å ha vært klassifisert som en IEC-PAS, «publicly available spesification», siden 2007 [35]), og ikke minst ble det vist til resultatene fra NAMUR-undersøkelsen som kom i samme tidsrom. NAMUR, en internasjonal brukerforening bestående av 121 medlemsselskaper innenfor kjemisk og farmasøytisk prosessindustri, kjørte en større feltundersøkelse av et WirelessHART-nettverk på BASFs anlegg i Ludwigshafen i Tyskland. Konklusjonen fra denne undersøkelsen var at teknologien tilfredsstiller foreningens krav til fleksibilitet, sikkerhet, robust ytelse, koeksistens og interoperabilitet for klasse «monitorering» [36].

Hvorvidt ISA100.11a vil passere IEC-standardisering og en eventuell tilsvarende undersøkelse som NAMUR-undersøkelsen gjenstår å se, men foreløpig ser vi ingen grunn til å tvile på det. Vi har heller ikke noe belegg for å mene at ISA100.11a er teknisk underlegen WirelessHART; kanskje heller tvert i mot med tanke på den brede applikasjonsprotokollstøtten ISA100.11a tilbyr. Vi mener imidlertid at WirelessHART er en mer moden protokoll, da den har rukket å være gjenstand for flere positive studier og vurderinger (blant annet utført av Statoil [27]), og det allerede finnes flere leverandører av WirelessHART-baserte, industrielle sensorprodukter. I tillegg har WirelessHART-standarden

en fordel ved at den lettere lar seg implementere grunnet dens mer spissede dekningsområde. På tross av dette finnes det for øyeblikket bare én produsent av selve brikkene: Dust Networks. Som nevnt i innledningen er dette en uheldig situasjon av flere årsaker, og derfor er det ønskelig å lage en åpen implementasjon av WirelessHART på en brikke fra en annen leverandør. I vårt tilfelle falt valget på en ZigBee-tiltenkt brikke fra produsenten Atmel, og både den og vår implementasjon av protokollen er beskrevet i senere kapitler. La oss imidlertid først se grundigere på WirelessHART-standarden.

4 WirelessHART

I dette kapitlet vil vi se nærmere på WirelessHART-protokollen [20] og noen fremtredende egenskaper ved denne. Vi tar hovedsaklig for oss datalinklaget da det er dette vi implementerer og som oppgaven kretser rundt. Datalinklaget er også det laveste laget som skiller seg helt fra IEEE 802.15.4-standarden. For å få en mer helhetlig oversikt vil vi gå gjennom hvordan WirelessHART forholder seg til IEEE 802.15.4-standarden, og også beskrive i korte trekk noen av egenskapene ved de øvrige lagene. Men først en kort beskrivelse av forhistorien til WirelessHART.

4.1 HART Communication Foundation

WirelessHART har sitt utspring i HART, den eldre og godt etablerte åpne kommunikasjonsprotokollen tilrettelagt for industrien. HART står for Higway Addressable Remote Transducer protocol, og styres av HART Communication Foundation (HCF). Protokollen er over 20 år gammel og det anslås at over 26 millioner enheter har protokollen installert [37]. HCF ble grunnlagt i 1993 og er organet som forvalter både HART og WirelessHART-protokollene. Organisasjonen er en non profit-organisasjon med over 200 medlemsbedrifter. Medlemmene nyter godt av brukerstøtte og opplæring i bruk av HART-utstyr. Det meste av dokumentasjonen rundt HART og WirelessHART kan bestilles fra HCF mot betaling, også for ikke-medlemmer.

HART er en relativ enkel og robust protokoll, både når det kommer til opprettelse av et HART-nettverk og drifting av dette nettverket. WirelessHART har som mål å videreføre disse egenskapene til den trådløse sensorteknologien [38]. Da WirelessHART-enheter også må støtte tradisjonelle HART-funksjoner, vil en organisasjon med et eksisterende trådbasert HART-nettverk som etablerer et WirelessHART sensornettverk unngå store kostnader, både ved opplæring av personell og til konfigurering av nytt utstyr.

4.2 Det fysiske laget

Som vi har vært inne på tidligere i teksten benytter WirelessHART seg av det fysiske laget beskrevet i IEEE 802.15.4-2006. Noen modifikasjoner har de dog gjort, og vi skal i dette kapitlet ta for oss disse. Se for øvrig kapittel 3.2.2 for mer detaljer om IEEE 802.15.4 PHY.

4.2.1 Generelle krav

Frekvensområde

IEEE-standarden kapittel 6.1.1 definerer tre modulasjonsvarianter i frekvensområdet 868/915 MHz i tillegg til 2,4 GHz. WirelessHART opererer kun innenfor 2,4 GHz frekvensområdet. (2400 til 2483,5 MHz).

Kanaltilordning

For 2,4 GHz-båndet tillater IEEE-standarden tilordning til kanalene 11 til 26. Tabell 4-1 under lister opp kanalnumrene og kanalenes midtfrekvens slik WirelessHART-spesifikasjonen beskriver. Kanal 26 er utelatt i WirelessHART da noen regioner ikke tillater bruk av denne kanalen.

Kanalnummer	Frekvens	Kanalnummer	Frekvens	Kanalnummer	Frekvens
11	2 405 MHz	**16**	2 430 MHz	**21**	2 455 MHz
12	2 410 MHz	**17**	2 435 MHz	**22**	2 460 MHz
13	2 415 MHz	**18**	2 440 MHz	**23**	2 465 MHz
14	2 420 MHz	**19**	2 445 MHz	**24**	2 470 MHz
15	2 425 MHz	**20**	2 450 MHz	**25**	2 475MHz

Tabell 4-1 Kanalnumre og frekvens

Tjenesteprimitiver

IEEE-standarden definerer et sett tjenester på det fysiske laget som datalinklaget kan benytte seg av. WirelessHART benytter ingen av disse nøyaktig slik de er definert i IEEE-standarden, men definerer i stedet selv et komplett og forenklet sett tjenesteprimitiver. Egenskapene til alle primitivene som beskrives skal implementeres, men det presiseres at en produsent/utvikler står fritt til å velge hvordan.

Det er skrevet mer om generell bruk av tjenesteprimitiver i kapittel 4.3.3 om tjenester i WirelessHART og kapittel 3.2.3 om tjenester i IEEE 802.15.4, og for eksempel i Figur 6-4 i implementeringskapitlet kan man se hvordan primitivene benyttes. Tjenesteprimitivene på det fysiske laget er listet opp under:

- *ENABLE.request*. Benyttes av laget over for å komme i kontakt med det fysiske laget og for å sette radioen i lytte- eller sendemodus.
- *ENABLE.confirm*. Blir gitt som svar på ENABLE.request, med en indikasjon på om det fysiske laget er i stand til å overføre data eller ikke.
- *ENABLE.indicate*. Det fysiske laget gir beskjed til laget over om at starten på en melding har kommet inn på radioen.

- *CCA.request*. Benyttes av laget over for å starte en CCA (Clear Channel Assesment).
- *CCA.confirm*. Blir gitt som svar på CCA.request over om status på utføringen av CCA-en. Mulige feil: TransceiverOff, ChannelBusy og ChannelIdle.

- *DATA.request.* Benyttes av laget over for å sende en datapakke (også ACK-pakker).
- *DATA.confirm*. Blir gitt som svar på DATA.request om status på utsendingen av datapakken. Mulige resultat: Success, TranceiverOff, ReceiverOn og TransmitterBusy.
- *DATA.indicate.* Det fysiske laget gir beskjed til laget over om at en (fullstendig) datapakke har kommet inn på radioen. Denne tjenesten sender med datapakken og signalstyrken pakken ble tatt i mot med.

- *ERROR.indicate.* Det fysiske laget gir beskjed til laget over om feil i mottaket av datapakke. Mulige feil: RecevierBufferOverflow og PacketIncomplete.

For å håndtere konfigurering av det fysiske laget tilbys det også tjenesteprimitiver av typen LOCAL_MANAGEMENT.request, LOCAL_MANAGEMENT.confirm og LOCAL_MANAGEMENT.indicate. Disse tar et parameter som refererer til hvilken tjeneste som tilbys. Tjenestene er:

- *RESET*. Initier det fysiske laget.
- *READ_TX_PWR_LEVEL*. Les av hvilken effekt (målt i dBm) radioen er satt til å sende med.
- *WRITE_TX_PWR_LEVEL*. Endre effekten (målt i dBm) radioen skal sende med.
- *WRITE_SLEEP_STATE* . Endre tilstanden til det fysiske laget. Tilstanden kan være ”sleep” eller ”awake”.

- *WRITE_RCV_OVERFLOW_ENABLE.* Slå av eller på muligheten til å indikere fullt mottakerbuffer (se ERROR.incicate over med RecevierBufferOverflow).

4.2.2 Konstanter og PIB-attributter

Konstanter

IEEE-standarden kapittel 6.4.1 definerer to konstanter som er avhengig av maskinvaren protokollen er implementert på:

- *aMaxPHYPacketSize.* Den største mulige PPDU-en (physical protocol data unit) en enhet kan motta skal være 127 oktetter i WirelessHART.
- *aTurnaroundTime.* Den lengste lovlige tiden det tar å snu radioen fra mottak til sending og omvendt skal være 12 symboler i WirelessHART (Se for øvrig om Tx til Rx og Rx til Tx under). Et symbol er et sample i den fysiske overføringen, og på 2,4 GHz tilsvarer dette 16 us i tid.

PIB-attributter

IEEE-standarden definerer attributter som er nødvendige for at det fysiske laget skal kunne håndteres av enheten, såkalte PIB-attributter (PAN Information Base). Disse kan i IEEE 802.15.4 leses, og i mange tilfeller endres, ved hjelp av administrasjonstjenestene som tilbys laget over. WirelessHART definerer imidlertid de fleste av disse attributtene som konstanter, og grensesnittet mot laget over kan derfor være enklere enn i IEEE-standarden. Grensesnittet består av implementeringen av primitivene beskrevet i avsnittet om tjenesteprimitver i kapittel 4.2.1, og vil bli nærmere gjennomgått senere.

4.2.3 868/915 MHz –båndene

IEEE-standarden beskriver tre forskjellige modulasjonsteknikker for 868/915 MHz-båndene. WirelessHART opererer kun på 2,4 GHz, som kun benytter seg av én av de tre modulasjonsteknikkene, O-QPSK, og WirelessHART-standarden tar naturlig nok ikke opp de utelatte teknikkene. O-QPSK (Offset Quadrature Phase-Shift Keying) håndteres av maskinvaren og vi går derfor ikke i detalj om dette, men hovedprinsippet er å utnytte faseskifting av signalbølgen til å gi høyere bitrate.

4.2.4 Krav til enhetens radio

Tx til Rx og Rx til Tx

IEEE-standarden definerer krav til hvor lang tid man maksimalt kan benytte for å snu retningen på radioen, relevant for eksempel i forbindelse med overgangen fra mottak av pakke til utsending av ACK. I begge tilfeller, altså fra Tx (sendemodus) til Rx (mottaksmodus) og fra Rx til Tx, er den maksimale tiden begrenset til konstanten aTurnaroundTime nevnt i avsnittet om konstanter i kapitlet over.

For øvrig beskrives Tx til Rx som tiden fra den siste biten tilhørende en utgående PPDU (Physical Protocol Data Unit) forlater radioen, til radioen er klar til å ta imot den første biten tilhørende en innkommende PPDU. Tilsvarende beskrives Rx til Tx som tiden fra den siste biten tilhørende en innkommet PPDU er mottatt på radioen, til radioen er klar til å sende ut den første biten tilhørende en utgående PPDU.

For WirelessHART definerer standarden et krav om at enheter skal kunne skifte mellom to kanaler i løpet av maksimalt 192 us (12 symboler). Det anbefales også at det ikke tar mer enn maksimalt 4 ms å skru på radioen fra avslått tilstand, uten at standarden begrunner dette noe mer.

Overføringseffekt

IEEE 802.15.4-standarden sier at alle radioer skal kunne sende med en EIRP (equivalent isotropic radiated power) på minst -3dBm. For WirelessHART er kravet at radioen må kunne gi EIRP 10 dBm ±3 dB, altså vesentlig mer. I begge tilfeller er dette snakk om krav til maksimalverdier. I praksis er det ønskelig å kjøre på lavere effekt for å unngå interferens med annet utstyr. Denne overføringseffekten skal kunne styres fra applikasjonen og radioen skal sette effekten til nærmeste mulige verdi. For øvrig skal produsenter av utstyr forholde seg til lovlige verdier gitt av lokale reguleringsorganer.

Sensitivitet

WirelessHART-standarden krever at alle enheters radio og antenne sammen skal oppnå en sensitivitet på 85 dB eller bedre. Sensitiviteten beskriver kort fortalt den laveste signalstyrken hvor enheten klarer å holde pakkefeilraten under et visst nivå, og vi tolker dette til at det menes at enhetens sensitivitet skal være -85 dBm (som er 85 dB ned fra 0 dBm).

Clear Channel Assesment (CCA)

IEEE 802.15.4-standarden krever at en enhet skal støtte minimum én av følgende CCA-teknikker:

- *CCA type 1 «energi over terskel»*. Overføringsmediet regnes som opptatt når enheten lytter og oppdager energi på det aktuelle frekvensområdet som er over ED–terskelen (energy detection terskel).
- *CCA type 2 «Carrier sense»*. Overføringsmediet regnes som opptatt når det oppdages et signal som identifiseres som IEEE 802.15.4-overføring på det aktuelle frekvensområdet, og som benytter samme modulerings- og transmisjonsteknikk som enheten selv benytter.
- *CCA type 3 «Carrier sense med energi over terskel»*. Overføringsmediet regnes som opptatt som i CCA type 1 og/eller CCA type 2. Med andre ord en kombinasjon av de to forrige CCA-teknikkene.

WirelessHART benytter seg kun av CCA type 2, noe som gjør at enhetene ikke har anledning til å undersøke om overføringsmediet er i bruk av andre teknologier (for eksempel WLAN – IEEE 802.11). Dette vil ha effekt på hvordan WirelessHART-enheter vil oppføre seg i sameksistens med andre teknologier på samme frekvens. Ved bruk av CCA type 3 vil for eksempel en pakke ikke sendes ut over radioen om det er aktivitet på frekvensen, og en ny overføring vil bli planlagt frem i tid. Slik WirelessHART fungerer (med CCA type 2) vil det oppstå pakketap i slike situasjoner, og retransmisjon må planlegges. I tid vil dette ikke bety noe, men vi antar at batteribruken øker når radioen benyttes til sending av pakker, sammenlignet med om enheten er klar over at frekvensen er i bruk. I det siste tilfellet kan enheten gå direkte til å planlegge neste forsendelse i stedet for å bruke energi på å overføre en pakke som uansett vil bli ødelagt.
For øvrig er det gjort forskning på sameksistens mellom WirelessHART og WLAN som konkluderer med at WirelessHART-nettverk har høy pålitelighet i miljøer hvor deler av frekvensområdet er i bruk [27].

4.2.5 Frekvens, modulering og transmisjon

Tabell 4-2 viser eneste lovlige frekvensområde, samt eneste modulerings- og transmisjonsteknikk, som skal benyttes i WirelessHART.

Teknologi	Krav
Transmisjonsteknikk	DSSS
Frekvensområde	2 400 – 2 483,5 MHz
Kommunikasjonsrate	2 000 kchip/s
Moduleringsteknikk	O-QPSK
Bitrate	250 kb/s
Symbolrate	62,5 ksymbol/s
Symboler	16-ary ortogonale

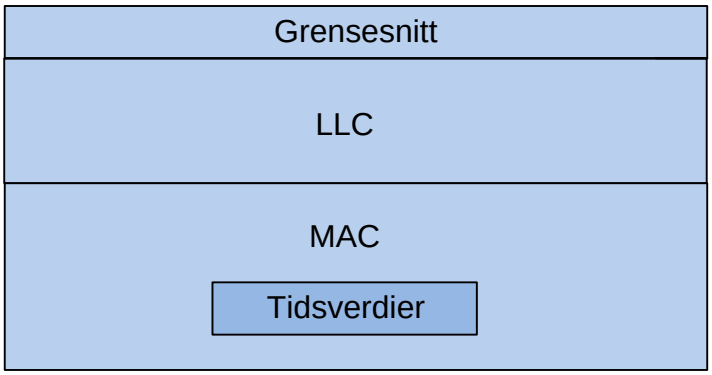
Tabell 4-2 Lovlige verdier

4.3 Datalinklaget

Datalinklaget fokuserer på overføring mellom to noder som er i direkte kontakt. Dette gjøres i WirelessHART ved hjelp av TDMA (Time Division Medium Access). I dette kapitlet skal vi se hvordan TDMA benyttes i WirelessHART, samt bli kjent med sentrale begreper som superrammer og linker og se på de ulike pakketypenes oppbygging. Det er viktig å være klar over at all kommunikasjon som omtales i dette kapittelet er begrenset til direkteoverføring mellom naboer, og at all eventuell ruting tas hånd om av nettverkslaget. Dette selv om også datalinklaget forholder seg til nettverksgrafer og nettverksadresser. Datalinklagets oppgave er i den sammenheng bare å identifisere og velge ut neste hopp.

4.3.1 Logisk oppdeling av datalinklaget

WirelessHART-standarden følger noenlunde samme oppdeling av datalinklaget som IEEE 802-protokollene. Den definerer et logisk linkkontroll-sublag (Logical Link Control – LLC) som blant annet definerer pakkestrukturen, flytkontroll og sikkerhetsmekanismer på datalinklaget, og et mediumtilgangskontroll-sublag (Medium Access Control – MAC) som tar seg av tilgangskontrollen til overføringsmediet. I tillegg defineres et grensesnitt opp mot nettverkslaget for tilgang til datalinklagets funksjonalitet, samt en rekke tidsverdier som kreves for gjennomføringen av mediumtilgangskontrollen. Figur 4-1 illustrerer denne oppdelingen av datlinklaget. Selv om vi så langt har gjennomgått protokollene fra bunnen og opp, finner vi det mest naturlig å begynne på toppen denne gangen. Dette fordi kunnskapen om bestanddelene i LLC-laget er hensiktsmessig ved den senere gjennomgangen av MAC-laget.



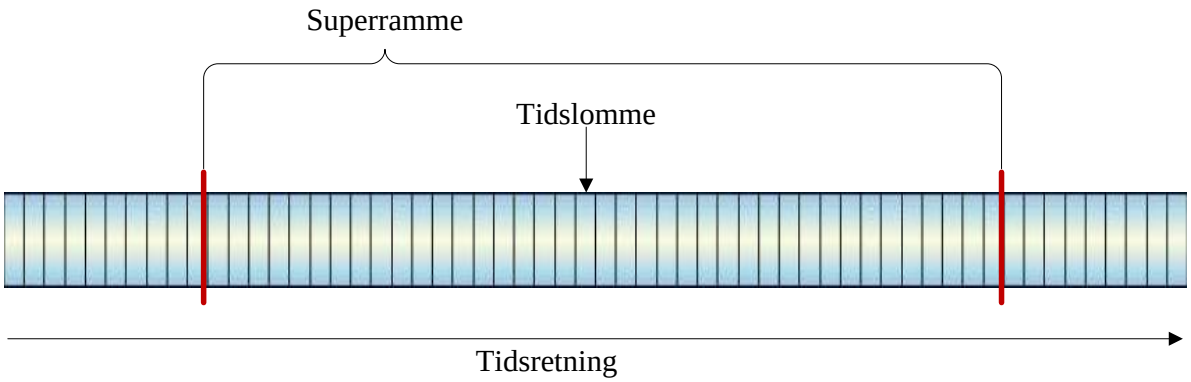
Figur 4-1 Logisk oppdeling av datalinklaget I WirelessHART

4.3.2 Generelt om datalinklaget

WirelessHART benytter Time Division Multiple Access (TDMA) samt kanalhopping på datalinklaget; to protokoller vi gikk gjennom i kapittel 3.1. TDMA gir determinisme og god kontroll på medietilgangen, mens kanalhopping tilbyr mulighet for å overføre data over flere kanaler samtidig i nettverket, og ikke minst god robusthet. Ved at to kommuniserende noder stadig skifter frekvens vil vedvarende støy på bestemte frekvenser ikke påvirke ytelsen til WirelessHART-nettverket nevneverdig.

Det er også mulig å svarteliste kanaler i WirelessHART. Kanaler som svartelistes benyttes da ikke til noen form for kommunikasjon i nettverket. Dette kan for eksempel være nyttig i tilfeller hvor en ekstern støykilde dominerer frekvensen.

TDMA i WirelessHART baserer seg på en eller flere parallelle superrammer som består av et antall tidslommer (time slots). Superrammene er avgrenset i tid, repeteres kontinuerlig, og kan være enten aktivert eller ikke. I superrammer som ikke er aktivert vil det aldri forekomme noen kommunikasjon mellom noder. Det må til enhver tid være minst en superramme aktivert, og alle enheter i nettverket må ha innebygget støtte for å håndtere flere superrammer.



Figur 4-2 En superramme med et sett tidslommer

Hver tidslomme kan tilordnes ett nodepar og disse nodene kan dermed utveksle data i løpet av denne tidslommen. Kommunikasjonen i en tidslomme er alltid enveis: En node er kilde, mens den andre tar imot. For toveis kommunikasjon mellom to noder A og B må man da nødvendigvis sette av (minst) to tidslommer: En med retning fra A til B, og en med retning fra B til A. Det er for øvrig ingenting i veien for å sette av flere tidslommer enn dette mellom samme nodepar i en superramme.

Når en pakke kommer frem til mottakernoden skal mottakernoden umiddelbart, og innen samme tidslomme, bekrefte mottaket ved at kommunikasjonen skifter retning og mottakeren returnerer en ACK. (Unntaket er om informasjonen som kommuniseres kringkastes. Da skal det ikke returneres noen ACK).

Kommunikasjonen mellom to noder foregår over en virtuell link, definert av en superramme, en tidslomme og et kanaloffset (mer om dette i kapittel 4.3.5). Ved at flere linker bærer data samtidig over forskjellige kanaler blir det muligheter for et stort antall linker ut fra hver node. Spesifikasjonen beskriver et teoretisk eksempel hvor man benytter 15 kanaler og hver superramme har 9000 tidslommer, og hvordan dette gir mulighet for 135 000 linker. Dette forutsetter at aktive linker i samme tidslomme operer på ulike kanaler. Til en tidslomme skal det kunne være tilknyttet én link til en annen node per superramme, og for hver superramme går noden gjennom en liste av linker og betjener alle linker hvor noden fungerer som en mottaker (i tilfelle noe skulle komme inn).

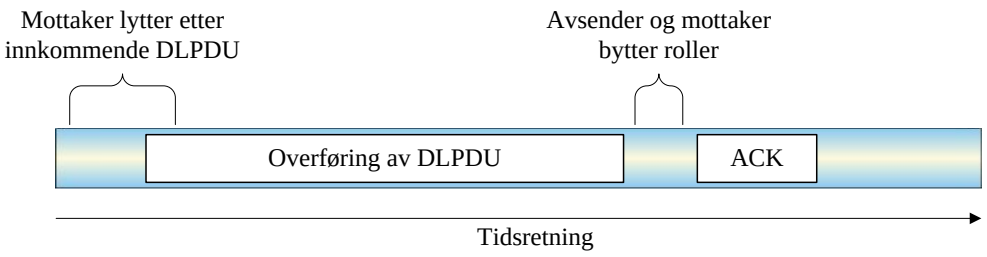
Klokkesynkronisering

Tiden som stilles til rådighet i en tidslomme er 10 millisekunder, og da endringer i det sensitive utstyret, for eksempel ved temperatursvingninger eller naturlig klokkedrift ikke er uvanlig [8], vil klokkesynkronisering over nettverket være avgjørende for at overføringene skal holde seg innenfor. Her finnes to teknikker som nodene benytter seg av, avhengig av hvilken type node det er snakk om. Når en mottaker får inn en datalinkpakke (DLPDU – Data Link Protocol Data Unit. Se kapittel 4.3.4 om LLC-sublaget under) beregnes en tidsdifferanse mellom når mottakeren mener at overføringen burde ha foregått og når den faktiske overføringen fant sted. Informasjon om den beregnede differansen blir sendt tilbake til avsender ved å legge den ved ACK DLPDU-en. (Denne informasjonen innhentes jevnlig av nettverksadministrator som har det overordnede ansvaret for tidssynkroniseringen). På denne måten vil alle ACK DLPDU-er bidra til å holde nodene synkronisert med nettverket. Den andre teknikken handler om nettverksklokken; i hver nodes oversikt over nabonoder finnes

det noen noder som er definert som enheter med et spesielt ansvar for synkronisering av nettverksklokken, såkalte tidskilder. DLPDU-er fra slike noder fører til at mottakende node synkroniserer nettverksklokken sin.

At tidssynkronisering er viktig ser vi ekstra tydelig hvis vi studerer tidslommen i noe mer detalj. I en tidslomme gis det et visst slingringsmonn for sender og mottaker til å stille inn kanalen, og å starte radioen (sende/lytte). En viss tid ut i tidslommen starter så senderen å overføre data. Mottakeren skal da allerede ha startet å lytte, for å tillate at en viss klokkedrift kan ha funnet sted. Etter overføringen skal mottakeren snu radioen fra lytting til sending og sende ut en ACK DLPDU. Den opprinnelige senderen skal da allerede ha snudd radioen fra sending til lytting. Om det oppstod problemer med mottaket av DLPDU-en, eller om den var korrupt, sendes det ikke ut noen ACK. WirelessHART benytter altså ikke noen negativ ACK (NACK), som visse andre overføringsprotokoller. En forenklet fremstilling av overføringen i en tidslomme er vist i Figur 4-3 under.

Det faktum at et sensornettverk kan være utsatt for stadige endringer gjør også at en node hele tiden må ha oppdatert informasjon om naboene sine. (For beskrivelse av lagret informasjon om en nabo, se eget underkapittel i 4.3.5). Dette oppfylles ved bruk av «keep alive»-DLPDU-er og «advertise»-DLPDU-er. «Keep alive» er til for å minne nærliggende noder om at en node fremdeles er i live, og «advertise» innbyr eventuelle nye noder til å bli med i nettverket. De forskjellige typene DLPDU-er er nærmere beskrevet i kapittel 4.3.4.



Figur 4-3 Overføring av en DLPDU i en tidslomme

Nodetyper

I IEEE 802.15.4-standarden beskrives to forskjellige nodetyper, FFD og RFD, hvor forskjellene hovedsaklig ligger i hva slags oppgaver nodene er i stand til å utføre (se kapittel 3.2 om IEEE 802.15.4). I WirelessHART er alle nodene i stand til å utføre de samme oppgavene, og en differensiering av noder er da ikke nødvendig. Alle nodene har støtte for både ruting, sinkruting og kilderuting. Alle pakker blir rutet gjennom nettverkslaget, og om

pakken ikke er kommet frem til destinasjonen sendt ned til datalinklaget og videre ut på en link.

4.3.3 Tjenester

Det fysiske laget tilbyr, som vist i kapittel 4.2.1, et sett av tjenester til datalinklaget. På samme måte tilbyr datalinklaget tilsvarende og andre tjenester til nettverkslaget over.
Disse tjenestene deles inn i meldings- og administrasjonstjenester. Vi skal i dette kapitlet se på hvordan de ulike meldings- og administrasjonstjenesteprimitivene er definert og benyttes for overføring av DLPDU-er og administrering av nodens datalinklag.
Tjenesteprimitiver er gjennomgått i kapittel 3.1.1, og kort oppsummert kan vi si at hovedvariantene henger logisk sammen som listen under viser:

- *Request.* Ber om å utføre en tjeneste.
- *Confirm.* Formidler resultatet av en tidligere request.
- *Indicate.* Indikerer en tjeneste som er utført.

De tre variantene er også å finne igjen i IEEE 802.15.4 standarden [23], og dermed også i ZigBee. I kapittel 3.1.1 definerte vi en fjerde hovedvariant, nemlig *response*, men den benyttes ikke i WirelessHART eller IEEE 802.15.4.

Meldingstjenester

Meldingstjenestene omhandler overføring av DLPDU-er over en logisk link mellom to naboer, og tjenesteprimitivene for dette deles inn i underkategoriene overførings-, nettverkshendelses- og mottagelsestjenesteprimitiver.
Primitivene knyttet til overføring er tilhørende TRANSMIT- og FLUSH-tjenestene. TRANSMIT.request blir initiert av nettverkslaget når det ønsker å sende en pakke, og kommer i fire varianter. Variasjonene består av forskjellige parametere avhengig av om pakken skal rutes via en graf, kringkastes eller bare leveres til en konkret adresse. Datalinklaget legger den i riktig kø for utgående pakker, finner den riktige overføringslinken som skal benyttes og innkapsler innholdet fra nettverkslaget i en DLPDU. (I IEEE 802.15.4 standarden finner man en ekvivalent tjenesteprimitiv kalt MCPS-DATA.request).

Når en TRANSMIT.request er utført sørger datalinklaget for å formidle utfallet av tjenesten til nettverkslaget ved hjelp av en TRANSMIT.confirm. (Tilsvarende primitiv i IEEE 802.15.4 er MCPS-DATA.confirm).
Når en DLPDU som er adressert til noden har ankommet blir dette formidlet til nettverkslaget ved å utstede en TRANSMIT.indicate. (Representert i IEEE 802.15.4 med MCPS-DATA.indication).
Det er mulig å fjerne en DLPDU som befinner seg i en utgående kø, og dette gjøres med primitiven FLUSH.request, hvis resultat gis ved FLUSH.confirm. (Disse primitivene har sine tilsvarende i IEEE 802.15.4 ved MCPS-PURGE.request og -confirm).
For nettverkshendelser er det fire tjenesteprimitiver, hvorav samtlige er indicate som sendes opp til nettverkslaget. Dette fordi det er nettverkslaget som har ansvaret for nettverkets topologier og nettverksadministrasjon. DISCONNECT.indicate utstedes når noden har mottatt en DLPDU fra en nabonode som forteller at den forlater nettverket. PATH_FAILURE.indicate utstedes når man ikke lenger får kontakt med en nabonode, og av dette antar at linken er nede. ADVERTISE.indicate utstedes når noden har mottatt en advertise DLPDU fra en nærliggende node som ønsker å bli en del av nettverket. NEIGHBOR.indicate utstedes når noden har tatt imot en DLPDU fra en annen node, og hvor denne andre noden ikke er å finne i nabolisten.
Ved å stille noden inn til konstant å lytte etter trafikk kan man få en oversikt over nettverkstrafikk og på denne måten diagnostisere egenskaper ved nettverket. Når noden er i denne modus kan den utstede en RECEIVE.indicate når den oppdager en DLPDU som ikke er adressert til denne noden.

Administrasjonstjenester

For konfigurasjon av datalinklaget og uthenting av statistikk benyttes tjenesteprimitivene LOCAL_MANAGEMENT.request, LOCAL_MANAGEMENT.confirm og LOCAL_MANAGEMENT.indicate. Via disse har man tilgang på 24 tjenester, avhengig av parameterne man sender med. Mange av disse tjenestene har sine korresponderende tjenesteprimitiver i IEEE 802.15.4, samtidig som en del også er unike for WirelessHART.
LOCAL_MANAGEMENT.request tar parameterene tjeneste og data. Tjeneste representerer funksjonaliteten man ønsker, mens data er en samling nødvendig informasjon for at tjenesten skal fungere som tiltenkt. Data-parameteret er ikke påkrevet av alle administrasjonstjenestene, men for de av dem som gjør det kan data ses på som tjenestens parametere.

De to andre primitivene, LOCAL_MANAGEMENT.confirm og LOCAL_MANAGEMENT.indicate, bærer i tillegg til tjeneste og data med seg argumentet status som skal representere resultatet av den aktuelle LOCAL_MANAGEMENT.request. (Se appendiks A for beskrivelse av de aktuelle administrasjonstjenestene).

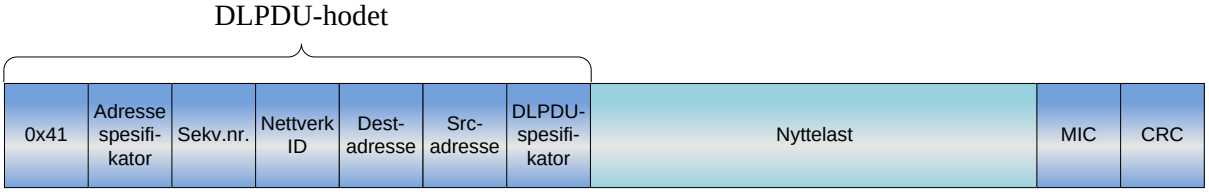
4.3.4 LLC sublaget

LLC-sublagets (logical link control) ansvar handler blant annet om å lage gyldige DLPDU-er, sørge for flytkontroll samt oppdagelse og håndtering av feil. Dette skal vi nå gå gjennom, og vi begynner med å beskrive en DLPDU generelt.

DLPDU (Data-Link Protocol Data Unit)

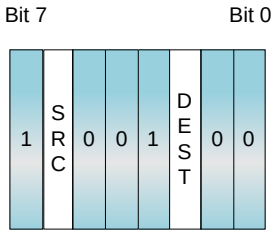
En DLPDU er pakken som kapsler inn en nettverkspakke (NPDU – Network protocol data unit) på vei ut på linken. I tillegg inneholder den en rekke kontrolldata som skal sørge for sikker og feilfri overføring til korrekt mottaker.
WirelessHART-standarden lister opp innholdet i en DLPDU:

- En byte (oktett) satt til verdien 41_{16}
- En byte som beskriver type adresser som skal benyttes
- Sekvensnummer representert ved en byte
- NettverksID representert med to byte
- Destinasjonsadresse (to eller åtte byte)
- Avsenderadresse (to eller åtte byte)
- En byte som beskriver DLPDU-ens type, prioritet og autentisitet
- Innholdet som skal kapsles inn
- Fire byte meldingsintegritetskode (MIC)
- To byte for feilsjekking



Figur 4-4 DLPDU-struktur med de ulike hodefeltene

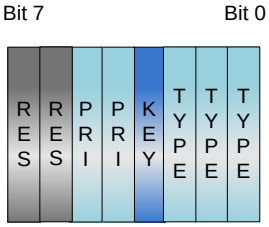
Alle felter som består av mer enn én byte (med unntak av nyttelasten) blir overført med minst signifikant byte først. Den første bytens (som vist i Figur 4-4) eneste formål er å beskrive starten på en DLPDU og skal alltid ha verdien 41_{16}. Bitene i den etterfølgende byten danner en adressespesifikator og beskriver hvorvidt man adresserer sender og mottaker med deres to-byte alternative adresse, eller den utvidete unike adressen på åtte byte. Kortversjonen defineres av nettverksadministratoren på applikasjonslagsnivå, og kan enten være et hvilket som helst tall mellom 0000_{16} og $FFFE_{16}$ eller kringkastingsadressen $FFFF_{16}$. Siden denne to-byte-adressen bare defineres innad i nettverket vil den ikke være globalt unik. Det vil derimot adressen som er bygd opp av åtte byte. Dette er en IEEE EUI-64 addresse, og denne bygges opp av en tre byte lang enhets-ID, en to-bytes kode for type enhet (tilordnet av Hart Communication Foundation) og tre byte som i WirelessHART alltid skal ha verdien $001B1E_{16}$ (tildelt Hart Communication Foundation av IEEE). Den korte adressen tildeles noden i det den har blitt en del av et nettverk, så den utvidede adressen brukes primært i «join»-fasen ved tilknytning av node til et nytt nettverk. Det er ingen avhengighet mellom avsender og mottaker på adresselengde, så enhver kombinasjon med kort og lang adresse kan benyttes. For å oppnå dette benytter man seg av bit nummer to (mottaker) og seks (avsender). Satt bit betyr lang adresse og ikke satt bit betyr kort adresse. De resterende bitene i adressespesifikatoren skal settes slik Figur 4-5 viser.



Figur 4-5 Adressespesifikator

Sekvensnummeret skal være identisk med den minst signifikante byten i ASN (Absolute Slot Number; en teller som holder rede på antall tidslommer passert siden nettverkets oppstart).
Enheter fra flere WirelessHART-nettverk kan befinne seg innenfor samme rekkevidde, og det er derfor nødvendig å kunne avgjøre nettverkstilhørighet. Dette gjøres ved å inspisere de neste to bytene. Disse inneholder en nettverks-ID som er et tall innenfor et definert tallområde. Det finnes fem forskjellige tallområder; ett for brukerdefinerte permanente nettverk, ett for

brukerdefinerte midlertidige nettverk, ett for nettverk brukt av produsenter og to som er reserverte.



Figur 4-6 DLPDU-spesifikator

Til slutt i hodet er det en DLPDU-spesifikator på én byte (Figur 4-6). Denne benyttes til å beskrive hvilken prioritet en DLPDU har og hva lags type den er. Spesifikatoren har også en bit (bit nummer tre) som indikerer hvorvidt en DLPDU går mellom enheter som er autentisert i nettverket eller ikke. Om denne biten ikke er satt betyr det at avsender er i ferd med å melde seg inn i nettverket og ikke har blitt autentisert ennå. De to mest signifikante bitene, bit seks og syv, er reservert og skal være satt til 0. Bitene fire og fem gjenspeiler en av prioritetsformene kommando, prosessdata, normal og alarm (se eget avsnitt nedenfor), og bitene null til to forteller hvilken type DLPDU det er snakk om. De fem mulige typene er ACK, advertise, keep-alive, frakobling og data, som alle er beskrevet nedenfor. Etter hodet kommer innholdet som man ønsker å sende; typisk en pakke fra nettverkslaget (NPDU) om det er en DLPDU av typen data, eller tidsberegninger for synkronisering om det er en ACK.
Til sist i DLPDU-en, som et endehode, kommer det fire byte med meldingsintegritetskoden (MIC – Message integrity code) og to byte for feilsjekking (CRC – Cyclic redundancy check). Meldingsintegritetskoden har som formål å forsikre at en DLPDU kommer fra en autentisert avsender innen nettverket, mens CRC benyttes til å oppdage bitfeil i pakken. Begge disse metodene er nærmere forklartet i et eget avsnitt lenger ned i dette kapitlet.
Pakkene på datalinklaget kan være av fem ulike typer: Data, ACK, keep-alive, advertise og frakobling. Vi skal i de neste avsnittene gå gjennom de viktigste egenskapene til disse DLPDU-typene.

Data-DLPDU

Som vi så i Figur 4-6 over benyttes de tre minst signifikante bitene til å beskrive pakketypen, og for data-DLPDU er verdien 111_2. Data-DLPDU er den eneste pakketypen hvis nyttelast i sin helhet går fra avsenders nettverkslag til mottakers nettverkslag. Nyttelasten leveres til avsendernodens datalinklag fra nettverkslaget, og sendes over linken som henvist av

nettverkslaget. Nettverkslaget i nabonoden som mottar pakken vil sørge for at den blir rutet videre ved behov.

ACK-DLPDU

ACK-DLPDU beskrives i DLPDU-spesifikatoren med verdien 000_2. Denne typen DLPDU, og resten av pakketypene, blir kun håndtert av datalinklaget.

Formålet til en ACK-DLPDU er å bekrefte (ACKnowledge) en overføring av en pakke fra en nabonode, samt gi tilbakmelding om hvorvidt pakken ble tatt i mot av mottakeren. Alle pakker som er adressert til den aktuelle noden, og som ikke selv er av typen ACK eller adressert til kringkastingsadressen, skal føre til en utsendelse av en ACK-DLPDU.

En ACK-DLPDU er også, som vi så i det tidligere underkapitlet om synkronisering, en viktig faktor i tidssynkroniseringen mellom nodene. Nyttelasten i en ACK vil derfor i tilegg til en responskode som beskriver utfallet av overføringen, være en to-bytes positiv eller negativ tidsverdi i mikrosekunder. Tidsverdien beskriver tidsdifferansen fra når pakken var forventet å komme inn på radioen til når den faktisk kom. En positiv verdi vil si at pakken kom inn tidligere enn forventet, og en negativ verdi vil si at pakken kom inn senere enn forventet.

Responskoden er én byte og kan bestå av verdiene 0 (feilfri overføring), 61 (ingen ledig plass i mottaksbuffer), 62 (ingen ledig plass i alarm-/hendelsebuffer) og 63 (for lav prioritet på pakken). Både kode 62 og 63 henger sammen med pakkenes ulike prioriteter som er nærmere forklart i eget avsnitt senere (side 51). WirelessHART har ingen egen NACK-DLPDU (Negative ACKnowledge), men benytter seg heller av de nevnte responskodene over. Ved alle andre situasjoner enn feilfri overføring blir pakken som har kommet inn forkastet.

Keep-Alive-DLPDU

Keep-alive-DLPDU-ens verdi i spesifikatoren er 010_2. Denne typen DLPDU sørger for at en link mellom to naboer holdes ved like selv om datakommunikasjonen mellom de to nodene er lav. Enheten vil også kunne oppdages av nye noder i nettverket ved å sende ut en keep-alive-DLPDU med jevne mellomrom.

Ved å sende ut en keep-alive-DLPDU vil man få tilbake en ACK som altså inneholder informasjon for tidssynkronisering.

Advertise-DLPDU

En advertise-DLPDU kjennetegnes av verdien 001_2 i spesifikatoren.

Formålet med denne typen pakker er å tilby tilstrekkelig informasjon for lyttende noder til at de kan koble seg til nettverket. For å koble seg til nettverket trenger en node en hel del informasjon, og nyttelasten i denne DLPDU-en kan bli omfattende.

Nyttelasten inneholder ASN (absolute slot number), kontrollmekanismer rundt innmelding i nettverket, hvilke sikkerhetsnivåer som støttes, en array som beskriver hvilke kanaler som er i bruk, ID for eventuelle rutinggrafer, informasjon rundt superrammer som er i bruk og informasjon om hver link den nye noden har fått tildelt. Den nye noden er bundet til disse «join»-linkene inntil nettverksadministratoren har godkjent noden og den er klar til å delta i nettverket som et fullverdig medlem.

Frakobling-DLPDU

Når noder har til hensikt å forlate et nettverk sender de ut en frakobling-DLPDU. Denne har verdien 011_2 i spesifikatoren.

DLPDU-en har ingen nyttelast, men nettverkslaget blir informert ved mottak av denne type pakke. Dette skyldes at nettverkslaget skal oppdatere rutingtabeller og annen nettverksinformasjon siden noden ikke lenger finnes i nettverket. Referanser til den aktuelle noden vil også på datalinklaget slettes i form av at alle linker til denne noden fjernes og noden tas ut av nabolisten.

Prioriteringer

I Figur 4-6 over så vi at to biter beskriver en prioritering. De mulige prioriteringene er kommando, prosessdata, normal og alarm i synkende prioritetsnivå. Verdiene er henholdsvis 11_2, 10_2, 01_2 og 00_2. Ved at noden vedlikeholder en prioriteringsterskel for hvilke pakker som skal tas imot og hvilke pakker som skal overses, kan trafikkflyten reguleres.

Det er verdt å merke seg at DLPDU-er av typene keep-alive, advertise og frakobling alltid skal aksepteres og føre til genereringen av en ACK, uavhengig av prioritet.

Blant annet for at nettverksadministratoren skal kunne regulere prioritetsterskelen, har alle DLPDU-er som påvirker administrasjonen av nettverket fått den høyeste graden av prioritet; kommando. Dette er pakker med diagnosedata, konfigurasjonsdata og kontrollinformasjon relatert til nettverksadministrasjon.

Det neste prioritetsnivået er prosessdata, og omfatter pakker med målinger fra sensorer og nettverksstatistikk. Pakker på dette prioritetsnivået skal tas imot av noden så sant køen for mottatte pakker er under 75% full.

Prioritetsnivået normal gjelder alle pakker som ikke omfattes av de to ovennevnte prioritetsnivåene eller prioritetsnivået alarm. Disse pakkene skal kun aksepteres av noden så lenge køen for mottatte pakker er under 50% full.
Det laveste prioritetsnivået er alarm, og gjelder for pakker som kun frakter alarm- eller hendelsesdata. Det kan virke forvirrende at «alarm» betegner det laveste prioritetsnivået, men det refererer til alarm- og hendelsesdata beskrevet i spesifikasjonens kapittel 8 hvor alarm får en annen betydning. En node skal aldri køe mer enn én pakke med dette prioritetsnivået, og eventuelle alarmpakker som kommer inn mens det ligger en annen alarmpakke i køen blir avvist.

Sikkerhet og feilhåndtering

LLC-sublaget har også ansvaret for sikkerheten og feilhåndtering på datalinklaget. Til dette benyttes henholdsvis MIC (Message Integrity Code) og CRC (Cyclic Redundancy Check). Vi summerer her kort opp de viktigste egenskapene i algoritmene knyttet til MIC og CRC som benyttes i WirelessHART.
MIC-en benyttes til autentisering av avsender, og kalkuleres ved hjelp av CCM* sammen med en AES-128 chiffer. Førstnevnte er en forkortelse for «Counter with CBC-MAC» hvor CBC-MAC igjen er en forkortelse for «Cipher Block Chaining Message Authentication Code». Stjernen markerer at dette er en modifisert variant av den opprinnelige CCM. Detaljene om CCM er åpent tilgjengelig i RFC 3610 [39], og vil ikke ble gjennomgått her, men vi skal kort se på hvordan CCM* benyttes i kombinasjon med et AES-128-chiffer i WirelessHART. AES er kort for «Advanced Encryption Standard», og AES-128 er varianten med 128 biters nøkkel. Et chiffer er en sekvens av hemmelige tegn som gjør ord og tekster uforståelige for andre enn dem som kjenner nøkkelen, og AES-algoritmen generer et slikt chiffer med fire strenger som parametere, som videre brukes til å generere MIC. Disse parameterne er:

- *a*. Data som skal autentiseres men ikke krypteres.
- *m*. Innholdet som skal krypteres.
- *N*. Et 13 byte stort «nonce» (tall som kun opptrer en gang).
- *K*. 128-bit AES-nøkkel.

Siden WirelessHART benytter MIC kun for autentisering, og DLPDU-en ikke skal krypteres, vil ’m’ ikke ha noe innhold, mens ’a’ vil bestå av DLPDU-en fra og med verdien 41_{16} til og

med nyttelasten. 'N' skal være 13 byte lang, og skal inneholde et tall som kun kan opptre én gang. Dette tallet er sammensetningen av ASN (som er fem byte) og avsenderadressen. ASN blir skrevet mest signifikante byte til N[0] opp til minst signifikante byte til N[4]. Som vi så i kapitlet om DLPDU (Data-Link Protocol Data Unit) kan en node ha to typer adresse. Om noden benytter seg av sin 8 byte EUI-64 adresse skrives dennes mest signifikante byte til N[5] opp til minst signifikant byte til N[12]. Benyttes derimot nodens to byte korte adresse skrives den mest signifikant byte til N[11] og minst signifikant byte til N[12]. Bytene N[5] til og med N[10] vil da bli satt til 0.

AES-nøkkelen i 'K' er en av to: Den offentlige nøkkelen (satt til 7777 772E 6861 7274 636F 6D6D 2E6F 7267_{16} for alle WirelessHART-enheter) eller nettverksnøkkelen. Den siste administreres av nettverksadministratoren og benyttes i all trafikk bortsett fra advertise-DLPDU-er og trafikk til og fra noder som ennå ikke er fullverdig del av nettverket.

En CRC beregnes over en DLPDU og kan benyttes til å oppdage feil i bitsekvensen. Avsender gjør en beregning basert på dataene som skal sendes og setter resultatet inn i CRC-feltet. Når en node mottar en DLPDU gjøres samme beregning over den mottatte pakken, og resultatet blir sjekket mot verdien i CRC-feltet på slutten av den aktuelle DLPDU-en. Om dette ikke er identisk blir DLPDU-en forkastet uten å gi beskjed til avsender. Om verdien stemmer kalkulerer mottakernoden en MIC og ser om denne stemmer med MIC-en i DLPDU-en. Også her blir DLPDU-en forkastet uten varsel til avsender om MIC-ene ikke stemmer overens.

4.3.5 MAC-sublaget

MAC-sublagets oppgave er i all hovedsak å sørge for at DLPDU-er transporteres feilfritt mellom to noder over en link. Til dette kreves det god kontroll på tidssynkronisering, fullstendig oversikt over naboer, linker og pakker som ligger i kø, og til slutt en mekanisme for valg av riktig link for overføringen. I dette kapitlet skal vi se nærmere på alle disse temaene, og vi starter med å se på overføringen i en tidslomme.

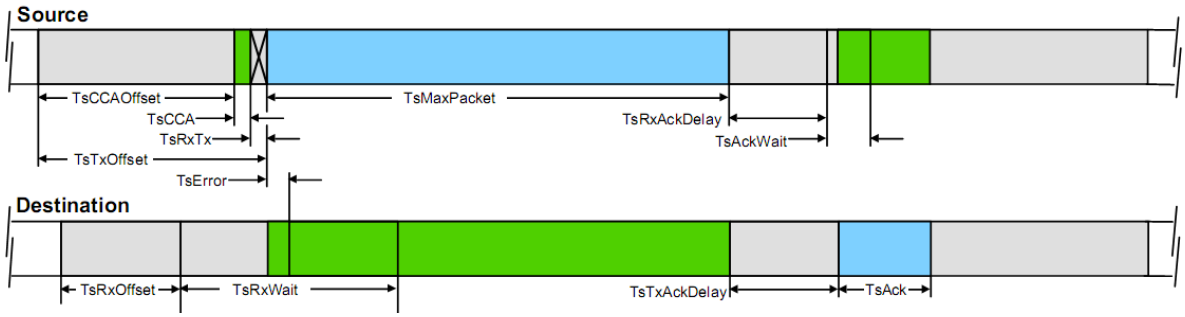
Synkronisering i tidslommer

Figur 4-7 viser henholdsvis avsender og mottaker i en overføring over en link. Linken er beskrevet av tidslommen i en superramme og de to nodene. Det er satt av tid for avsender til å klargjøre pakken og å sjekke om det er støy på kanalen, så overføringen starter ikke umiddelbart i begynnelsen av tidslommen. Mottakeren venter en avsatt tid før den starter å lytte. Det er viktig at mottakeren lytter etter den innkommende pakken i tiden før og etter

forventet ankomst. Dette slingringsmonnet gjør overføring mulig også der mindre forskyvninger i klokkene har funnet sted. Nodene vil som tidligere omtalt justere klokkene sine i etterkant ved tilbakemelding om for sen eller for tidlig avsending av en pakke fra en node som er en tidskilde.

I de fleste overføringer vil det være forventet at en ACK blir returnert, og det er satt av tid til dette i samme tidslomme som overføringen av selve datapakken. Etter at avsender har sendt datapakken ut på linken, skifter altså radioen fra sende- til lyttemodus. På mottakersiden skifter nødvendigvis radioen fra lytte- til sendemodus. Tiden mellom ferdig utført overføring av datapakken til begynnelsen av overføringen av en ACK har mottaker til rådighet for kontroll av mottatt datapakke, utregning av tidsdifferanser og generering av en ACK DLPDU. Til slutt i tidslommen er det satt av tid for nodene til prosessering av de mottatte pakkene (ACK for avsenderen og datapakken for mottakeren) samt planlegging av neste overføring.

Tidsintervallene er vist i Figur 4-7 (figuren er hentet uredigert fra standardens side 44), og beskrevet i Tabell 4-3 under.



Figur 4-7 Tidsintervaller i en tidslomme for både sender og mottaker ved overføring

Navn	Beskrivelse av tidsintervallet	Verdi (µs)
TsTxOffset	Før eventuell kanalsjekk	2 120
TsRxOffset	Før mottaker skal starte lytting	1 120 (± 100)
TsRxWait	Tiden satt av til å lytte etter meldingsstart	2 200 (± 100)
TsMaxPacket	Sending av maksimal pakkelengde (133 Byte)	4 256
TsTxAckDelay	Fra mottatt pakke til start sending av ACK	1 000 (± 100)
TsRxAckDelay	Fra utsendt pakke til start lytting etter ACK	800 (± 100)
TsAckWait	Tiden satt av til å lytte etter ACK	400 (± 100)

TsAck	Sending av maksimal ACK-lengde (26 Byte)	832
TsCCAOffset	Starten av tidslommen til start på CCA	1 800 (± 100)
TsCCA	Utføring av CCA (8 symboler)	128
TsRxTx	Skifte modus på radioen (Tx til Rx / Rx til Tx)	192

Tabell 4-3 Tidsintervaller i en tidslomme hvor det kan forekomme en overføring

Kommunikasjonsinformasjon lagret i noden

Alle nodene må til enhver tid ha tilgang til den nødvendige informasjonen for å kommunisere med sine naboer. Dette inkluderer en oversikt over alle nabonoder, superrammer, grafer samt hvilke tidslommer noden er tilordnet. Vi skal her se hvilke krav WirelessHART-spesifikasjonen stiller på disse punktene.

Alle noder er nødt til å innfri et minstekrav til lagringskapasitet og oversikt rundt kommunikasjonsinformasjonen. Denne informasjonen er gitt i tabell 4 på side 47 i standarden og er gjengitt i Tabell 4-4 under.

Kommunikasjonsvariabel	Minimumskrav
Naboer	32
Superrammer	16
Linker	64
Grafer	32
Graf til nabo - koblinger	128
Utgående pakker	16

Tabell 4-4 Minimumskrav og kommunikasjonsvariabler i en node

Naboer

Som vist i tabellen over må hver node ha kapasitet til å lagre informasjon om minimum 32 nabonoder. Det er ikke et krav at alle nodene skal være tilknyttet denne noden via en link; det er tilstrekkelig at denne noden har overhørt en kommunikasjon fra en annen node for å lagre informasjon om den. Informasjonen som skal lagres er følgende:

- Nabonodens unike ID, typisk den lange adressen
- Nabonodens korte adresse

- Innmeldingsprioritet (tilordnet av nettverksadministratoren)
- Hvorvidt naboen er en tidskilde
- Back-off-eksponent til bruk i kollisjonsunngåelsesalgoritmen (CA)
- Back-off-nedtellingsvariabel til bruk i kollisjonsunngåelsesalgoritmen
- Tidspunkt for siste kommunikasjon med denne naboen
- Nedtelling før linken regnes som død. (Startes på nytt for hver overføring)
- Gjennomsnittlig signalnivå på kommunikasjonen med naboen (måles i dBm)
- Antall pakker sendt til naboen (ikke kringkastingspakker)
- Antall avsendte pakker hvor det ikke er registrert mottatt ACK
- Antall pakker mottatt fra naboen (ikke kringkastingspakker)
- Antall kringkastingspakker mottatt fra naboen

Standarden stiller ikke krav til at alle disse verdiene skal lagres fysisk i nodene. Tatt i betraktning sensornoders lave lagringskapasitet generelt, spesifiseres det at en del av verdiene lar seg regne ut i sanntid, og derfor ikke behøver ta opp verdifull lagringsplass. Dette er med på å tilrettelegge for at noder fra forskjellige leverandører kan implementere forskjellige løsninger.

Superrammer

Som vi så vidt har nevnt tidligere skal en node kunne håndtere flere parallelle superrammer. Hvordan dette fungerer i praksis er nærmere omtalt i kapittel 6.4, og illustrert i Figur 6-6. Minimumskravet for hvor mange superrammer en node skal kunne håndtere er som vist i Tabell 4-4 16 superrammer. Alle superrammene trenger ikke være i bruk til enhver tid da nettverksadministratoren kan aktivere og deaktivere superrammer. I tillegg er en superramme ikke aktiv når den blir initiert. Informasjon noden trenger om superrammen er:

- Superrammens ID
- Superrammens størrelse (målt i antall tidslommer)
- Hvorvidt superrammen er aktiv eller ikke
- Liste med linker tilhørende superrammen

Linker

WirelessHART-standarden definerer en link som «full communication specification between adjacent devices in the network […]». Det betyr at en link i denne sammenheng er en samling av informasjon om hvordan to spesifikke nabonoder skal kommunisere. All denne informasjonen konfigureres av nettverksadministratoren, og består hovedsaklig av informasjon om nabonoden som utgjør «andre enden» av linken, hvilken tidslomme i superrammen linken er satt opp til å benytte, samt et kanaloffset for fortløpende å kunne avgjøre hvilken kanal som skal brukes. Det er bare rom for én overføring (bestående av DLPDU pluss eventuell ACK) per tidslomme, og en link har derfor også en definert kommunikasjonsretning. Enten er det en sendelink, en mottakslink, eller en delt link. Sistnevnte er en variant av sendelink som kan benyttes av flere noder, i motsetning til en vanlig sendelink som er dedikert, og muligheten for kollisjoner er derfor til stede. I slike tilfeller, der aktivitet på kanalen oppdages, vil en back-off-periode bli beregnet, og noden vil ikke forsøke å sende på den delte linken igjen før back-off-perioden er ute (mer om dette i avsnittet om linkfordeleren nedenfor), altså en variant av CSMA/CA.

Siden linker er retningsbestemte, må minst to linker opprettes mellom to naboer for å oppnå toveiskommunikasjon. Det er heller ingenting i veien for å opprette flere linker mellom to naboer i samme superramme hvis det er behov for hyppigere kommunikasjon.

En link kan være en av fire typer - normal, kringkasting, join eller discovery. En link mellom to nabonoder hvor begge er fullverdige medlemmer av nettverket betegnes som normal. En link hvor mottaker er definert som en gruppe noder (og hvor DLPDU-en blir adressert med kringkastingsadressen) betegnes som en kringkastingslink. Når en node er i ferd med å koble seg til nettverket opprettes det linker mellom den nye noden og noden den kommuniserer med, og linken som benyttes frem til den nye noden blir godtatt i nettverket kalles join-link. Denne har begrenset funksjonalitet siden noden ikke er verifisert av en nettverksadministrator ennå. Det blir også satt opp linker forbeholdt lytting etter nye noder, såkalte discovery-linker.

Listen under viser hvilke egenskaper ved en link noden skal ha kunnskap om.

- Linkens ID
- Referanse til aktuell nabonode eller kringkastingsadresse
- Type link
- Hvorvidt linken benyttes til mottak eller sending
- Hvorvidt linken er delt med flere noder

- Nummer på tidslommen i superrammen
- Kanaloffset

Grafer

Det er i WirelessHART mulig å foreta ruting ved hjelp av grafer. En graf er en retningsbestemt liste av stier som knytter sammen to enheter i nettverket, og blir konfigurert av nettverksadministratoren med en global identifikator. I noden som er pakkens avsender registrerer man pakkens mottakeradresse og hvilken graf man skal benytte. Dette skjer i høyere lag, og formålet med tilgang til informasjon om grafen så langt ned som datalinklaget er for å ha oversikt over hvilke naboer som kan fungere som neste hopp i en graf, og dermed fortløpende kunne avgjøre hvilken link som til enhver tid bør benyttes. Informasjonen datalinklaget trenger tilgang til er en graf-ID og en liste over naboer som kan benyttes for neste hopp i denne grafen. Utover dette skal en node ha muligheten til å inneha informasjon om adressene til pakkens endelige mål. Dette er ikke obligatorisk, men muligheten må være der siden det kreves for alle noder som kilderuter (sourcer) at de innehar adressen til den noden de kilderuter mot. Den nødvendige informasjonen er følgende:

- Grafens ID
- Endenodens lange adresse
- Endenodens korte adresse
- Liste med naboer som kan benyttes

Vi ser at det altså ikke er nødvendig med en full oversikt over grafene på datalinklaget, siden datalinklaget kun konsentrerer seg om neste hopp.

Midlertidig lagring av pakker

Pakker fra nettverkslaget (administrasjonspakker eller utgående pakker) må kunne lagres midlertidig i noden, med andre ord køes i en utgående kø på datalinklaget, og standarden beskriver informasjonen som må lagres som listen under:

- Pakkeidentifikator
- Nyttelast
- Overføringsprioritet

- Mottakerinformasjon
- Tidspunkt for når pakken ble lagret

Standarden sier ikke hvor store ressurser som skal settes av til slik mellomlagring, kun at en node skal kunne køe flere samtidige tjenesteprimitiver av typen TRANSMIT.request som genererer utgående DLPDU-er (Data-Link Protocol Data Unit). Merk at det ikke er selve den utgående DLPDU som mellomlagres, men alle bestanddelene som er nødvendige for å konstruere DLPDU-en ved sendetidspunktet.
Når en tjenesteprimitiv av typen confirm utstedes ovenfor nettverkslaget er formålet å formidle resultatet av en tidligere utført tjeneste av typen request. For å knytte de to tjenestene til hverandre sendes det med en identifikator (handle), og det virker naturlig at denne kan utgjøre pakkeidentifikatoren i forbindelse med mellomlagring av pakker.
Pakkens prioritet er den samme som omtalt i eget underkapittel tidligere, og har altså en av verdiene kommando, prosessdata, normal eller alarm. I forbindelse med køingen av pakker i noden er prioriteten nyttig i de tilfeller hvor to pakker konkurrerer om samme tidslomme. I disse tilfellene er det pakken med høyest prioritet som blir behandlet først, eller eventuelt den eldste pakken dersom pakkenes prioritet er lik.
På nettverksnivå tilbyr WirelessHART graf- og kilderuting (se kapittel 4.4.1 for flere detaljer). Dette er informasjon som påvirker valget av link på datalinklaget når pakken skal sendes ut. Om pakkens mottaker er en nabonode sendes pakken over en av linkene dedikert til sending til denne nabonoden. Om mottakeren er en node som er i ferd med å bli medlem av nettverket sendes pakken over en link spesifisert til kommunikasjon med slike noder, en såkalt join-link. Om mottakerinformasjonen er kringkastingsadressen benyttes linker satt av til kringkasting. I de tilfellene hvor ruting foregår ved grafruting vil mottakerinformasjonen bestå av en graf-ID. Som nevnt tidligere skal datalinklaget ha oversikt over hvilke naboer som kan benyttes for hver graf, og må derfor velge ut en link som går til en av naboene assosiert med denne grafen gitt i mottakerinformasjonen. Utvelgelse av egnet link og tidslomme er for alle tilfellene noe som håndteres av linkfordeleren.
Tidspunket for lagringen av pakken benyttes for å fjerne pakker som har ligget for lenge på køen. Dette vil normalt være pakker som av ulike årsaker ikke har nådd mottakeren og derfor fremdeles ligger på køen med tanke på retransmisjon.

Linkfordeler (Link scheduler)

Linkfordeleren har som oppgave å avgjøre og planlegge hvilken link som skal behandles til hvilken tid, og er derfor en svært sentral del av datalinklaget. Vi vil i kapittel 0 gå grundig gjennom denne i forbindelse med vår egen tolkning og implementering av en WirelessHART-linkfordeler, og nøyer oss derfor med en overordnet gjennomgang av spesifikasjonen og de teoretiske sidene ved linkfordeleren her.

For å avgjøre hvilken link som er den neste til å betjenes må noden stadig beregne hvilken tidslomme i de ulike aktive superrammene den befinner seg i, samt gå gjennom oversikten over linker i disse superrammene for å finne ut av hvilken link som har en tidslomme som ligger kortest frem i tid for håndtering. Hvilken tidslomme i superrammen som er den inneværende kan man finne ved å beregne ASN (absolute slot number) modulo antall tidslommer i superrammen, og dermed er det en grei sak å sammenlikne dette tallet med de ulike linkenes dedikerte tidslommenummer for å finne hvilke som vil inntreffe først. Normalt vil den av disse linkene som ligger nærmest frem i tid, og som enten er en mottakslink eller som er en sendelink med ventende pakker på køen, bli valgt ut. Det kan imidlertid fra tid til annen være flere linker satt opp på samme tidspunkt (samme ASN) hvis det er flere aktive superrammer i nettverket, og i så tilfelle prioriteres linkene etter retning: Sendelinker som har ventende utgående data går foran mottakslinker. Er det flere sendelinker på tidslommen som har ventende pakker velges den linken hvor neste pakke har høyest prioritet. Er det fortsatt flere kandidater velges den av pakkene som er adressert til den naboen det er lengst tid siden det forrige gang ble kommunisert med.

For mottakslinker er saken noe enklere: Alle tidslommer som er satt av til å lytte etter pakker skal betjenes med mindre det også er en pakke å sende i en sendelink i samme tidslomme – da prioriteres alltid sendelinken. Skulle to mottakslinker ønske behandling i samme tidslomme prioriteres den som tilhører den superrammen med lavest ID-nummer.

Disse nevnte mekanismene gjelder for vanlige linker adressert til en spesifisert nabo og for kringkastingslinker. I tillegg finnes det også noen spesielle linker som krever en litt annen behandling, så vidt nevnt i avsnittet om linker tidligere i kapittelet: delte linker, join-linker og discovery-linker.

Delte linker, altså at flere noder har muligheter til å sende til samme mottaker i samme tidslomme, medfører at pakker kan kollidere. Dette vil føre til retransmisjon av pakker, og for at retransmisjonen ikke skal føre til atter en kollisjon behøves en mekanisme for å velge når pakkene skal retransmitteres. Her kommer back-off-nedtellingsvariabelen og -eksponenten til

nytte, som så vidt ble omtalt i avsnittet om naboer tidligere. Ved å ta utgangspunkt i back-off-eksponenten (BOExp) til den aktuelle delte linken, som i utgangspunktet er initialisert til 0, kan et sett mulige verdier til back-off-nedtellingsvariabelen (BOCntr) beregnes:

$$Mulige\ verdier = 2BOExp - 1$$

Ved kollisjon tilordnes BOCntr en tilfeldig verdi blant settet av mulige verdier og BOExp økes med 1, og for hver delte link mot naboen man forsøker å sende en pakke til reduseres BOCntr med 1. Ikke før BOCntr har verdien 0 kan det foretas et nytt forsøk på overføring.
Pakketapet kan, foruten kollisjon, også være forårsaket av annen støy på frekvensen, og om en senere overføring til den samme naboen over en direkte (ikke delt) link også feiler skal linken regnes som nede. I slike tilfeller skal både BOExp og BOCntr nullstilles.
Join-linker er en variant av delte linker, og behandles deretter. Forskjellen er at disse linkene er reservert for nye enheter som forsøker å bli en del av nettverket, og benyttes av disse enhetene i en «join-periode» før de har blitt autentisert og fått tildelt dedikerte linker.
Discovery-linker benyttes for å legge til rette for at nye noder kan bli en del av nettverket, og for at noder allerede i nettverket skal identifisere hverandre som naboer. Det må settes av minst en av disse linkene i hver superramme, og de deles av alle nodene i nettverket. I utgangspunktet er discovery-linkene mottakslinker hvor nodene lytter etter nye naboer, men med tilfeldige mellomrom (en varierende tidsperiode mellom 0 og verdien «DiscoveryInterval» satt av nettverksadministratoren) skal enhetene også sende ut en keep-alive-DLPDU, slik at de lyttende naboene kan oppdage noden hvis de ikke allerede har gjort det. Både lyttingen og den tilfeldige utsendingen av keep-alive-DLPDU-er er linkfordelerens ansvar.
Et siste ansvarsområde for linkfordeleren er å sørge for utsending av advertise-pakker (se advertise-DLPDU i kapittel 4.3.4) hos noder som er konfigurert som advertise-utsendere av nettverksadministratoren. Avgjørelsen om når slike pakker skal sendes ut styres gjennom en tidsnedteller. Når tellerens verdi er 0 skal en advertise-pakke sendes ut på første utlink som ikke er i bruk, og som ikke er en kringkastingslink.

4.3.6 Logisk oppdeling av MAC-sublaget

For at MAC-sublagets oppgaver skal utføres korrekt, må det til enhver tid holdes oversikt over tidspunkter og hvilken tilstand noden er i. Til dette beskriver WirelessHART-

spesifikasjonen en TDMA-maskin og en sende- og mottaksmotor (Henholdsvis XMIT- og RECV-motor). I dette kapitlet vil vi gå gjennom disse tre bestanddelene som er svært sentrale for MAC-sublaget i WirelessHART.

TDMA-maskinen

For å hjelpe til med mulige handlinger datalinklaget kan utføre, sørger TDMA-maskinen for å opprettholde en tilstand. Når noden er i en viss tilstand er det regler for hvilke tilstander som kan være den neste. For eksempel kan en node aldri få tilstanden «Join» etter at den har blitt medlem av nettverket. Ei heller kan en node gå direkte fra tilstanden «Talk» til «Listen».

Tilsammen seks tilstander er mulige i TDMA-maskinen: De nevnte «Join», «Talk» og «Listen», samt «Idle», «Wait for ACK» og «Answer».

TDMA-maskinen «starter» ved at en node blir en del av et nettverk (tilstand «Join»). Etter at den har konfigurert superrammer, grafer og linker er noden klar til å ta imot og sende DLPDU-er. Tilstanden vil da være «Idle». De øvrige tilstandene vil bli nærmere omtalt i de følgende avsnittene, før vi til slutt i Figur 4-8 viser hvordan endring i tilstand og pakkeoverføringshendelser henger sammen.

Spesifikasjonen definerer TDMA-maskinen til å ha ansvaret for de følgende tre MAC-oppgavene:

- Opprettholde planlagte oppgaver
- Overføring av en DLPDU (både sende og motta).
- Sørge for tidssynkronisering.

Opprettholde planlagte oppgaver

For at linkplanlegging skal fungere er man avhengig av oppdaterte og korrekte superrammer og linker. Under opprettholdelse av planlagte oppgaver ligger derfor opprettelse og administrering av superrammer og linker, samt statistikk over nabonoder.

Så lenge en node er i tilstanden «Idle» er det fire situasjoner som kan oppstå, og som alle skal resultere i en ny beregning av neste link som skal betjenes, altså en ny linkfordeling. Den vanligste av disse fire situasjonene er at tidslommen til en planlagt link inntreffer og skal behandles. Om det er en sendelink blir tilstanden satt til «Talk», og om det er en link for mottak blir tilstanden satt til «Listen».

De resterende tre situasjonene i «Idle» endrer ikke tilstanden. Dette er tjenesteprimitivene FLUSH.request og TRANSMIT.request (se 4.3.3), samt oppdatering av enhetens superramme eller linker.

Overføring av DLPDU

Når en tidslomme med en planlagt utlink inntreffer, vil den tilhørende ventende pakke bli forsøkt sendt. Om enheten sender en DLPDU til kringkastingsadressen regnes overføringen som feilfri så snart pakken har forlatt radioen, siden slike pakker ikke skal bekreftes med ACK. Pakken kan slettes fra datalinklaget og tilstanden settes til «Idle». Overføringen regnes også som feilfri om det, for en DLPDU som er adressert til en nabo, mottas en ACK med en responskode 0 (se 4.3.4 om ACK DLPDU). Når pakken har forlatt radioen går avsender fra tilstand «Talk» til «Wait for ACK», og så snart overføringen er å regne som feilfri kan pakken slettes fra datalinklaget og tilstanden blir satt til «Idle».

Det er hovedsakelig to måter feilaktige overføringer kan oppdages på, som begge bør resultere i planlegging av retransmisjon før tilstanden settes til «Idle». Den ene måten er at det mottas en ACK med en responskode som indikerer at pakken er mottatt, men ikke akseptert. Den andre er når det ikke kommer en ACK i det hele tatt. Om det i det siste tilfellet ble benyttet en delt link vil BOExp og BOCntr beregnes før retransmisjon til denne naboen planlegges på nytt, som omtalt i avsnittet om linkfordeleren på side 60.

Mottak av en pakke forekommer i tidslommer med innkommende link. Når slike tidslommer opptrer går tilstanden fra «Idle» til «Listen» for å ta imot pakken. Om det ikke kommer noen pakke beregnes neste link og tilstanden blir satt tilbake til «Idle». Når en pakke blir mottatt er det tre muligheter: Pakken er adressert til noden og den blir tatt imot, noden er et punkt på veien og pakken skal videresendes, eller pakken er ikke adressert til noden i det hele tatt. I det pakken har kommet inn beregnes differansen mellom forventet tidspunkt for innkommet pakke og faktisk tidspunkt (TsError), og om pakken kommer fra en av tidskildene (omtalt i 4.4.4) til enheten blir nødvendig tidssynkronisering utført. Om pakken kommer fra en nabo som ikke er blant enhetens tidskilder blir det ikke gjort mer hva gjelder tidssynkronisering.

Når pakken har kommet inn må noden vurdere hva som skal gjøres med pakken. Ut fra pakkens prioritering, enhetens prioriteringsterskel (satt av nettverksadministratoren) og ledig kapasitet blir pakken enten godkjent eller avvist. Denne vurderingen skjer mens noden fremdeles er i tilstanden «Listen».

Om pakken ikke blir godkjent og den ikke var adressert til kringkastingsadressen blir det opprettet en ACK-DLPDU med en responskode som beskriver hvorfor pakken ikke ble godtatt. Den innkommende pakken blir avvist og enheten blir satt til tilstanden «Answer» for å sende ut ACK-en.

Om den innkomne pakken er en kringkastingspakke settes tilstanden rett til «Idle» siden pakken ikke skal bekreftes med ACK. Laget over blir varslet via en TRANSMIT.indicate hvis pakken blir godtatt og ikke forkastet på grunn av prioritet eller kapasitet.

I de tilfellene hvor pakken er adressert til noden og blir godtatt, blir det opprettet en ACK-DLPDU med responskoden «Success», og laget over blir varslet via en TRANSMIT.indicate. Deretter blir tilstand satt til «Answer» for å sende ut ACK-en. Den eventuelle ACK DLPDU-en skal ha med seg informasjon om tidsdifferansen mellom forventet mottak av pakke og faktisk mottak, samt en responskode. Når ACK-en er sendt ut blir neste link planlagt og nodens tilstand satt tilbake til «Idle».

Overføringsmotoren (XMIT)

Ansvaret for å sende en klargjort DLPDU ut på radioen overlates (logisk sett) til en egen overføringsmotor kalt XMIT-motoren. Denne støtter overføring med og uten bruk av CCA (Clear Channel Assessment). Når CCA skal benyttes settes først radioen i mottakermodus ved hjelp av primitiven ENABLE.request, før CCA.request blir utført. Når CCA-en er utført kan radioen settes i sendemodus.

Om det ikke skal benyttes CCA settes radioen rett i sendemodus, også ved hjelp av ENABLE.request. Selve overføringen starter ved at motoren utsteder primitiven DATA.request etter at tidsvariabelen TxDelay har gått ut. Når pakken er sendt venter XMIT-motoren på primitiven DATA.confirm fra det fysiske laget. Denne indikerer at alt gikk greit og motoren kan indikere videre ovenfor TDMA-maskinen at overføringen var vellykket.

Mottaksmotoren (RECV)

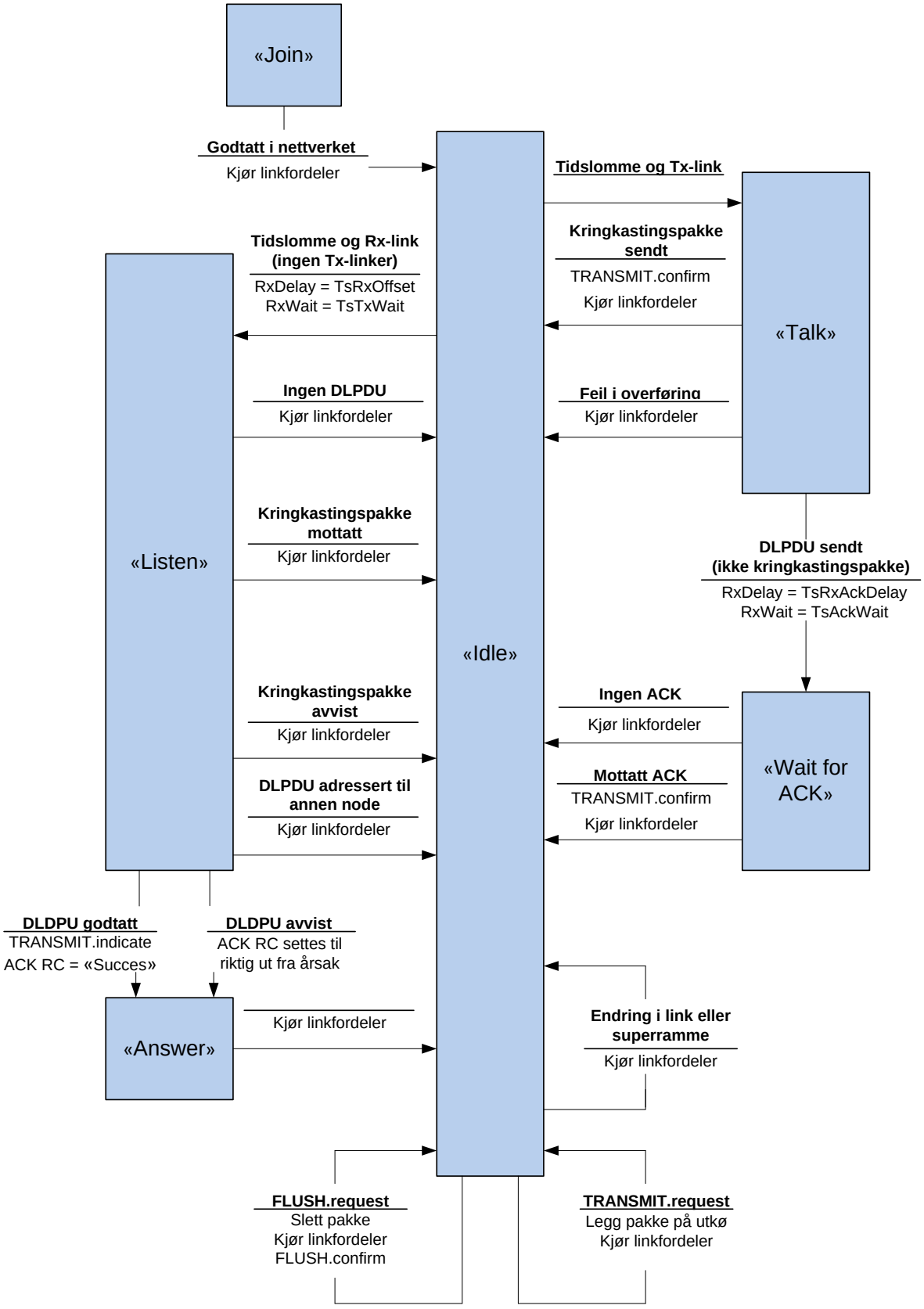
Standarden beskriver også en logisk enhet for selve mottaket av en DLPDU: Mottaksmotoren RECV, som har ansvaret for selve mottaket av pakken.

Som for XMIT-motoren benyttes det primitiver mot det fysiske laget. ENABLE.request blir benyttet for å sette radioen i mottakermodus, og når tidsverdien RxDelay går ut blir radioen bedt om å lytte etter data på den kanalen som ble angitt i ENABLE.request. RECV-motoren sørger også for å ta vare på tidspunktet for når pakken kom inn, til beregning av tidsdifferanse

for synkronisering. Om det ikke kommer noen pakke innen tidsvariabelen TsRxWait går ut vil RECV-motoren avslutte og gi TDMA-maskinen beskjed om dette.

Motoren vet når hele pakken er kommet inn ved at den mottar primitiven DATA.indicate fra det fysiske laget, og den kan da foreta en vurdering av pakkens adressering, CRC og MIC. Om pakken ikke er adressert til noden, eller CRC og/eller MIC ikke stemmer, avslutter motoren og gir TDMA-maskinen beskjed som om det ikke har kommet inn noen pakke. Hvis det kun er feil på CRC blir nodens statistikk over kommunikasjon oppdatert.

Om pakken er riktig adressert, og både CRC og MIC stemmer, blir pakken godkjent og RECV-motoren gir TDMA-maskinen beskjed om innkommet pakke. Ansvaret for den videre behandlingen overføres til TDMA-maskinen, som skifter tilstand i henhold til resultatet fra RECV-motoren og fortsetter med neste steg slik Figur 4-8 viser. Dette kan for eksempel være å sende ut en ACK, eller å gå direkte til planlegging av neste link.



Figur 4-8 Tilstandsmaskinen med hendelser som fører til endringer i tilstanden

Verdier mot det fysiske laget

For at datalinklaget, og spesielt TDMA-maskinen, skal fungere som man ønsker er det nødvendig at man har tilgang til visse verdier som egentlig hører til på det fysiske laget. Vi så for eksempel hvordan XMIT og RECV-motorene forholdt seg til tidsverdier rundt radioens egenskaper, og også hvordan motorene satte i gang timere rundt disse verdiene. Tabell 4-3 i underkapittel 4.3.5 viser hvilke tidsverdier datalinklaget er avhengig av å kjenne til.

4.4 Andre lag i WirelessHART

Frem til nå har vi gått gjennom det fysiske laget og datalinklaget i WirelessHART nokså detaljert. Det er flere årsaker til at vi har lagt hovedvekten på disse to lagene: For det første er det disse lagene det er naturlig å starte med ved oppbyggingen av en protokoll. For det andre er datalinklaget trolig det mest omfattende av lagene i WirelessHART-protokollen, og vi anså det som usannsynlig at vi ville rekke over noe mer i denne omgang. En WirelessHART-implementering er likevel ikke fullstendig med mindre man også implementerer lagene over datalinklaget: Nettverkslaget, transportlaget og applikasjonslaget (se for øvrig Tabell 3-1). I dette kapitlet vil vi gå gjennom disse lagene i et forsøk på å gi leseren et helhetlig bilde av WirelessHART. Gjennomgangen er altså mindre detaljert enn gjennomgangen av det fysiske laget og datalinklaget.

4.4.1 Nettverkslaget

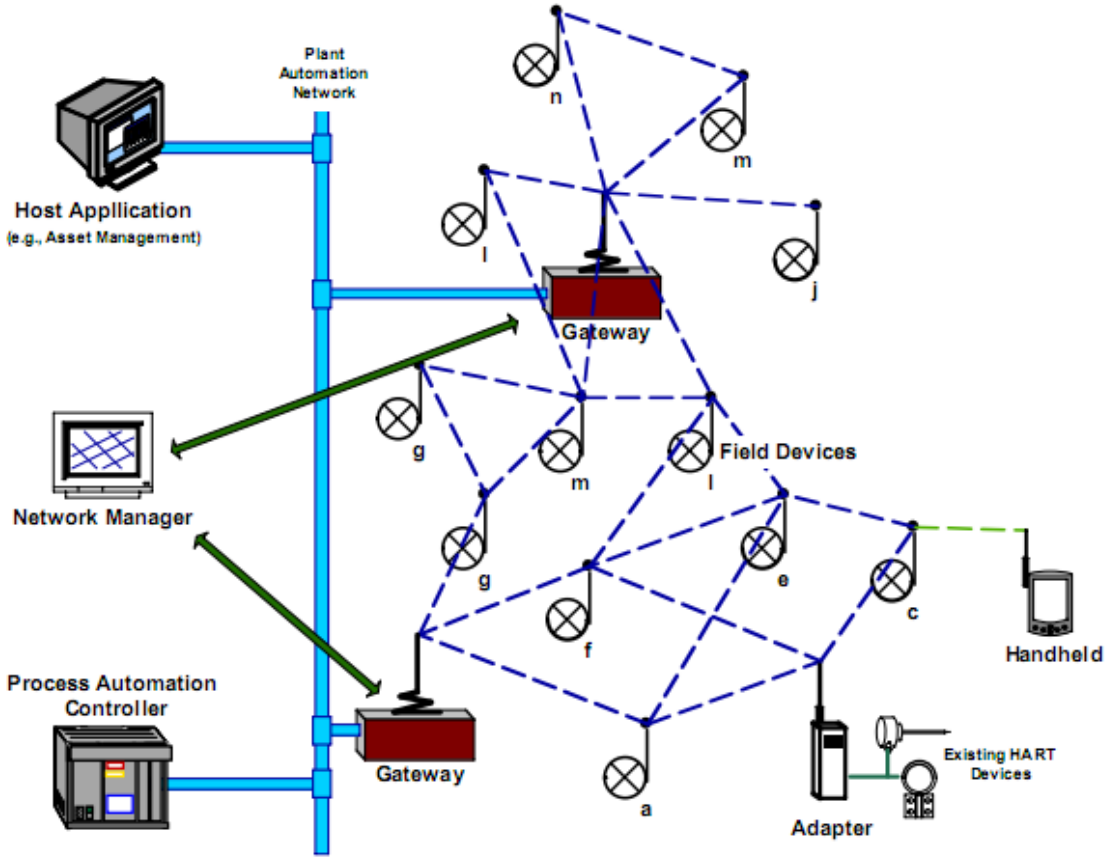
Nettverkslaget har det overordnede ansvaret for å håndtere ruting, nærmere bestemt grafruting og kilderuting (source routing), i nettverket. Dette er begreper vi allerede har vært innom i datalinklaget siden vi til en viss grad forholder oss til dem også der, men hovedsaklig er dette en del av nettverkslaget og vil bli nærmere gjennomgått senere i dette kapitlet.

Nettverkslaget er strukturert på samme måte som de to underliggende lagene: Øverst ligger et sett med tjenester som tilgjengeliggjøres for de overliggende lagene. Dernest følger den interne logikken og en rekke konstanter, attributter og datastrukturer som muliggjør rutingmekanismene og sørger for at enheten fungerer i nettverket. En del av disse dataene er delt med datalinklaget siden de er av relevans for begge to. Ved dataoverføring konstruerer nettverkslaget en egen pakketype (NPDU – Network Layer Protocol Data Unit) med tilstrekkelig informasjon til at pakken kommer frem til mottakernoden. Siden dette følger det samme mønsteret som lagene vi allerede har gått gjennom, og siden vi har valgt å fokusere hovedsaklig på datalinklaget, går vi ikke i mer detalj på dette i denne omgang. I stedet skal vi

raskt gå gjennom noen elementer som er spesielt for nettverkslaget, og som forhåpentligvis kan bidra til den totale forståelsen av protokollen. Vi starter med å se på nettverkets enkelte bestanddeler.

Nettverkets bestanddeler

Et WirelessHART-nettverk består av vanlige noder, også kalt feltnoder (field devices), et tilknytnignspunkt (gateway) som muliggjør kommunikasjon mellom feltnodene og applikasjonen som kjører på nettverket, og en nettverksadministrator som har som oppgave å konfigurere nettverket.



Figur 4-9 Typisk WirelessHART-nettverk [20]

Feltnodenes formål er å rapportere målinger de er programmert til å utføre inn til kontrollapplikasjoner. Alle feltnoder skal dessuten være i stand til å rute pakker fra andre noder. Sammenlignet med en node i et rent IEEE 802.15.4-nettverk (beskrevet i kapittel 3.2.1) er altså alle nodene i et WirelessHART-nettverk en FFD (Full Function Device).

Figur 4-9 er hentet fra WirelessHART-standarden og illustrerer enhetene i nettverket og hvordan de er koblet til en fabrikks hovednettverk.

Topologi

WirelessHART-noder er konfigurert for å støtte maskenettverk, men kan i praksis være del av nettverk av forskjellige topologier. Det kan være et stjernenettverk, hvor alle nodene er ett hopp fra tilknytningspunktet, eller det kan være et maskenettverk med flere alternative stier. I et WirelessHART-maskenettverk vil opprettelse av nettverket samt innføring av nye enheter være relativt enkelt for sluttbrukeren da nettverket formes dynamisk ut fra nettverkadministratorenhetens instruksjoner. Så lenge et tilknytningspunkt og en nettverksadministrator er konfigurert og alle nodene som vil knytte seg til nettverket har fått programmert nødvendig informasjon (join-nøkkel, nettverks-ID m.m) vil det være tilstrekkelig å plassere ut noder hvor det er ønskelig og nettverket skal i utgangspunktet kunne etablere seg selv. Det vil også reparere seg selv om deler av nettverket går ned.

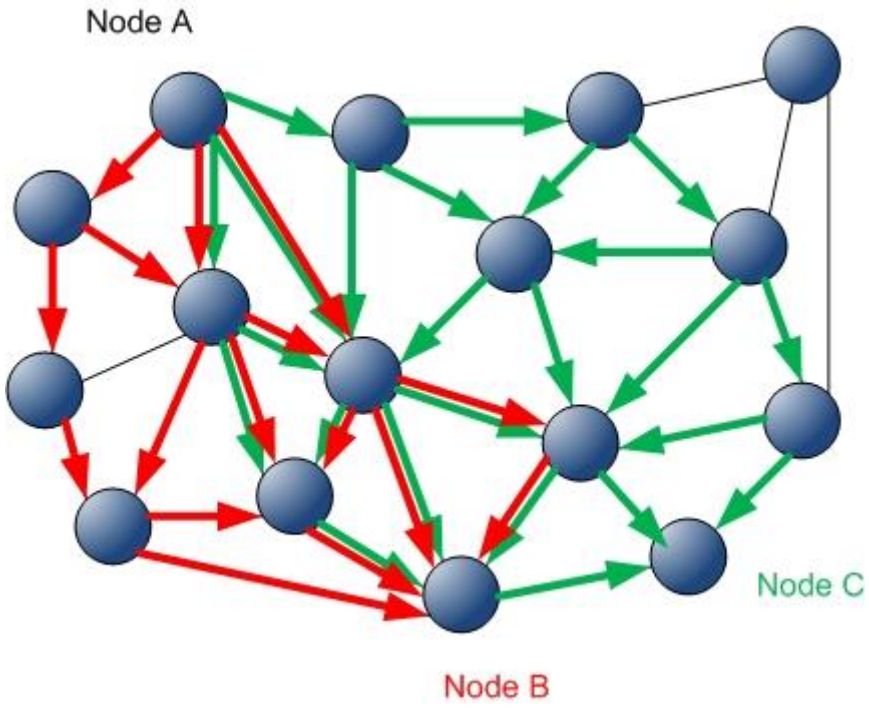
Med bruk av maskenettverk vil man som regel konfigurere nettverket til å ha redundans i mulige veier til en kilde. Så selv om en node for eksempel er kun ett hopp fra tilknytningspunktet og nettverksadministratoren vil det finnes alternative veier (gitt at det også er andre noder som kun er ett hopp fra tilknytningspunktet). Linken som foretrekkes når det er konfigurert flere alternativer er den naboen med sterkest signal.

Ruting

Ruting foregår på nettverkslaget og WirelessHART støtter to fremgangsmåter for å rute pakker: grafruting og kilderuting (source routing).

En graf representerer veien fra en avsender til én bestemt mottaker. Den er rettet, uten noen forekomster av løkker, og består av flere alternative stier underveis. Den eneste informasjon om mottaker som sendes med en pakke er grafens unike ID. Med andre ord er det én graf per mottaker, og graf-ID fungerer derfor som en sluttadresse i dataoverføringen. Dette gjør også at alle enheter som er del av rutene grafen representerer må inneha informasjon om grafen. De fleste noder er del av flere grafer og vi husker fra kapittel 4.3.5 at nodene måtte inneha grafinformasjon på datalinklaget. Denne informasjonen dreier seg kun om neste hopp på veien fra denne noden, og består av en liste med mulige neste hopp for hver graf. I følge standarden er grafruting svært pålitelig og tilbyr høy redundans, og det oppfordres til å benytte denne type ruting for det meste av datatrafikken.

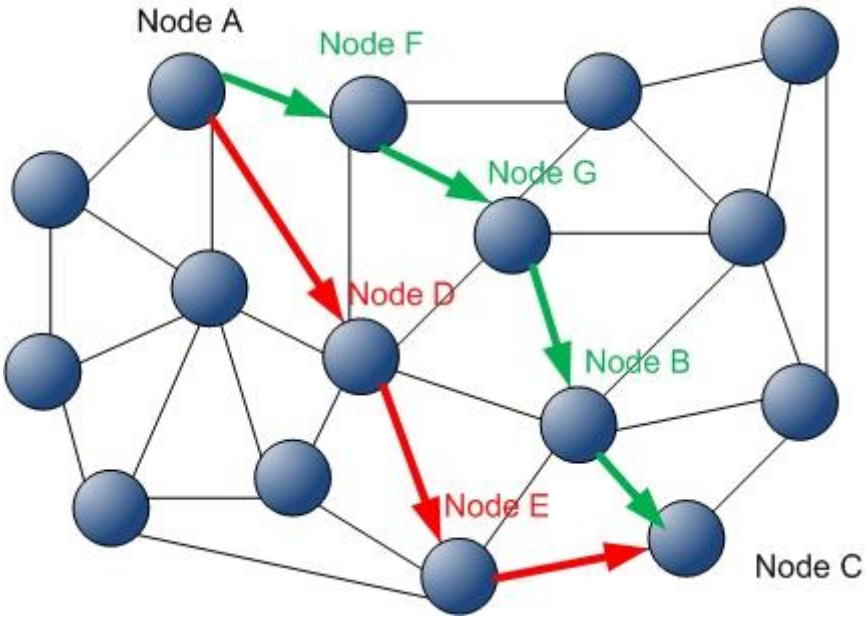
I Figur 4-10 har vi vist to grafer som node A benytter for å sende pakker til node B (rød graf) og node C (grønn graf).



Figur 4-10 Grafruting

Ved kilderuting blir informasjon om alle nodene på veien til mottakernoden lagt i pakken og det er ingen valgmuligheter langs ruten som benyttes. Når en node mottar en kilderutet pakke inspiseres riktig felt i pakkehodet og pakken videresendes til neste node på veien. Siden det ikke er noe redundans i kilderuting vil pakker gå tapt om deler av veien er ødelagt. Kilderuting er altså på langt nær så pålitelig som grafruting, men er nyttig når man vil teste kvaliteten på en spesifikk rute og annen nettverksdiagnostikk.

Figur 4-11 illustrerer to ruter fra en avsender (node A) til en mottaker (node C) ved bruk av kilderuting. Om en pakke skal rutes via den røde ruten vil pakken beskrive nodene mellom node A og node C som nodene D og E. Om pakken derimot skal rutes via den grønne ruten vil pakken beskrive nodene mellom A og C som F, G og B.



Figur 4-11 Kilderuting

4.4.2 Transportlaget

Transportlaget som kan benyttes i WirelessHART er ikke en obligatorisk egenskap, men kan benyttes til å sørge for fullendt overføring av data fra en node til en annen i nettverket. Gjerne over flere hopp, i motsetning til datalinklaget som kun har ansvar for neste hopp.

Transportlaget kan benytte seg av bekreftelse av oversendingen ved hjelp av ACK-meldinger, eller det kan også la være. Ved overføring hvor man ikke benytter ACK har man ingen garanti for at pakken kommer frem. Denne type overføring kan være hensiktsmessig for pakker som skal rapportere inn målinger. Disse vet man blir sendt med jevne mellomrom, så en retransmisjon kan derfor i mange tilfeller være unødvendig da neste pakke uansett sannsynligvis vil komme innen rimelig tid.

Om man konfigurerer transportlaget til å benytte seg av overføring som krever ACK blir det gjort klart en tunnel mellom de to nodene, og avsender venter med å sende neste pakke til en ACK som bekrefter forrige pakke har kommet tilbake. Å benytte ACK fra ende til ende i en kommunikasjon er foretrukket når man for eksempel skal konfigurere en enhet og har behov for å vite om konfigureringen gikk bra, eller når en node rapporterer hendelser (for eksempel at en ny node er oppdaget og skal innlemmes i nettverket).

For øvrig har transportlaget også en egen pakkestruktur (TPDU - Transport Layer Protocol Data Unit) som blir nyttelasten i pakken på nettverkslaget (NPDU-en). TPDU-en har to byte

som bestemmer egenskapene til pakken og en nyttelast som utgjøres av en eller flere applikasjonslagskommandoer.

4.4.3 Applikasjonslaget

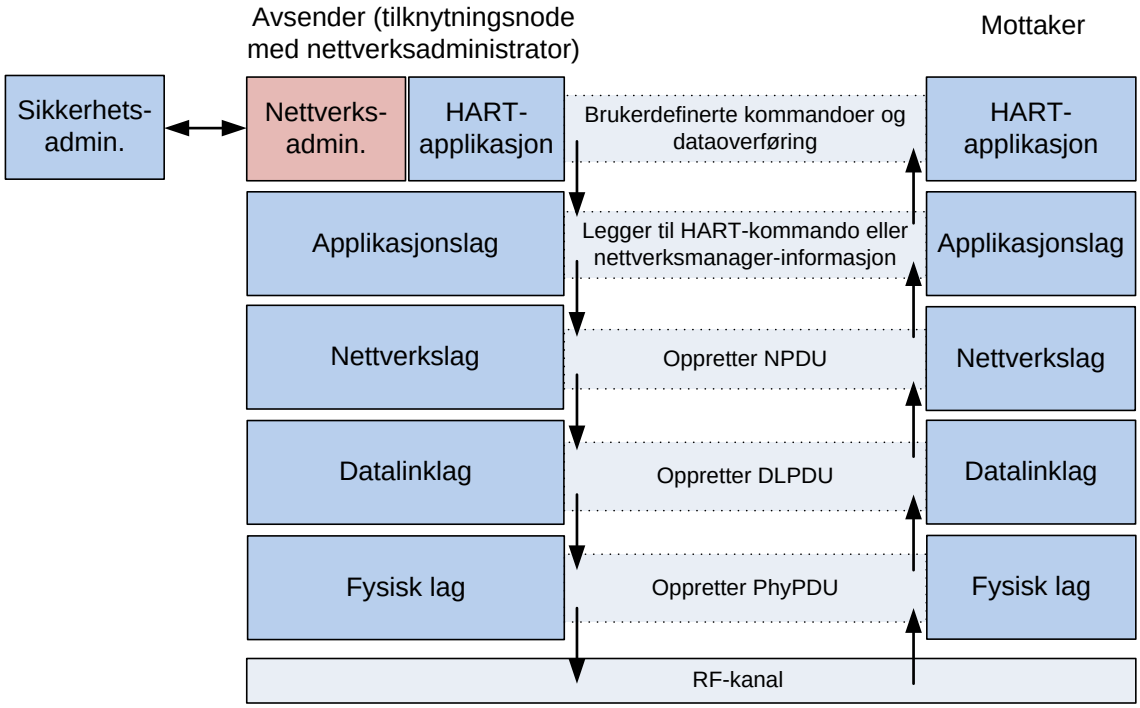
Vi har sett at WirelessHART er en trådløs adaptering av den trådbaserte protokollen HART, og WirelessHART kan kun benytte seg av et HART applikasjonslag. Dette definerer kommandoer, responser, datatyper og statusrapporteringer mellom enheter som HART-protokollen støtter. Alle enheter som er kompatible med HART, og da også WirelessHART-enheter, skal implementere alle kommandoer fra IEC-standardene IEC 61158-5-20 [40] og IEC 61158-6-20 [41]. Dette er et minimum av kommandoer, og det anbefales å implementere flere. Det spesifiseres at kommandoene skal implementeres nøyaktig slik de blir beskrevet i de to IEC-standardene, sammen med de endringer og tilleggskommandoer som er beskrevet i WirelessHART-standarden.

All kommunikasjon på applikasjonslaget foregår via kommandoer, enten det gjelder konfigurering av en enhet til å sende målinger til faste tider, eller om man ønsker at en enhet skal slå mikrokontrolleren av og på (tilsvarer å skru av og på strømmen til enheten). Hovedsakelig finnes det fire typer kommandoer:

- *Universelle kommandoer* – kommandoer som er knyttet til grunnleggende funksjoner som alle feltenheter må implementere.
- *«Common Practice» kommandoer* – gir tilgang til vanlig men ikke obligatorisk funksjonalitet implementert i feltenhentene.
- *Enhetsspesifikke kommandoer* – egenutviklede kommandoer som er spesifikke for akkurat denne type feltenhet. Disse kommandoene er beskrevet av enhetens produsent, og Hart Communication Foundation (HCF) er ikke involvert i utviklingen.
- *Gruppebegrensede kommandoer* – kommandoer rettet mot en gruppe feltenheter. Gruppen defineres av enhetens egenskaper som for eksempel temperatursensor eller trykksensor o.l. Denne type kommandoer gir tilgang til enheters spesifikke egenskaper uten å gå via enhetsspesifikke kommandoer.

4.4.4 Nettverksadministrator

Nettverksadministratoren er programvare som har som oppgave å konfigurere og håndtere nettverket. Spesifikasjonen sier ingenting om hvor nettverksadministratoren skal være plassert, men den skal ha en sikker kommunikasjonskanal både til tilknytningsnoden (gatewayen) og en sikkerhetsadministrator, så plassering i enten en tilknytningsnode eller på fastnettsiden av WirelessHART-nettverket vil være fornuftig. (Sikkerhetsadministratoren er en applikasjon som bidrar med generering og oppbevaring av sikkerhetsnøkler. Sikkerhetsadministratoren kan være ansvarlig for flere nettverk samtidig, og er ikke en del av protokollstakken). Figur 4-11 viser hvordan flyten går fra nettverksadministratoren til en mottakende node.



Figur 4-12 Flyten gjennom lagene i WirelessHART

Det er altså nettverksadministratoren som former nettverket. Den holder en fullstendig oversikt over alle enhetene i nettverket og sørger for at de har en unik kortadresse (16-bits «nickname»). Et viktig poeng å ta med seg er at nettverksadministratoren bestemmer hvilke tidslommer som tilordnes hvilke linker, og at det er den som overvåker nettverkets tilstand ved å hente inn statistikk fra de enkelte feltenhetene. En viktig faktor i et WirelessHART-nettverk er tidssynkroniseringen, og det er også nettverksadministratoren som bestemmer hvilke noder som skal fungere som tidskilder på datalinklaget. Tiden som skal synkroniseres

tar utgangspunkt i tiden i tilknytningsnoden, og ved at tidskilder blir oppdatert utover fra denne vil hele nettverket etter hvert bli synkronisert.

Det er mulig å installere mer enn en nettverksadministrator i nettverket for å oppnå redundans, men det skal aldri være mer enn en som er aktiv om gangen.

Når nettverket startes opp (eller når nye noder knyttes til nettverket) annonserer tilknytningspunktet eller allerede inkluderte noder at nettverket eksisterer. Når flere og flere noder kobler seg til bygger nettverksadministratoren stier i nettverket og strukturer disse til grafer i maskenettverkstopologien, for så å planlegge kommunikasjon ved for eksempel å tilordne tidslommer til kommuniserende noder. Hver gang endringer forekommer i nettverket må kommunikasjonsdata oppdateres og distribueres til nodene i nettverket.

5 Utviklingsmiljø

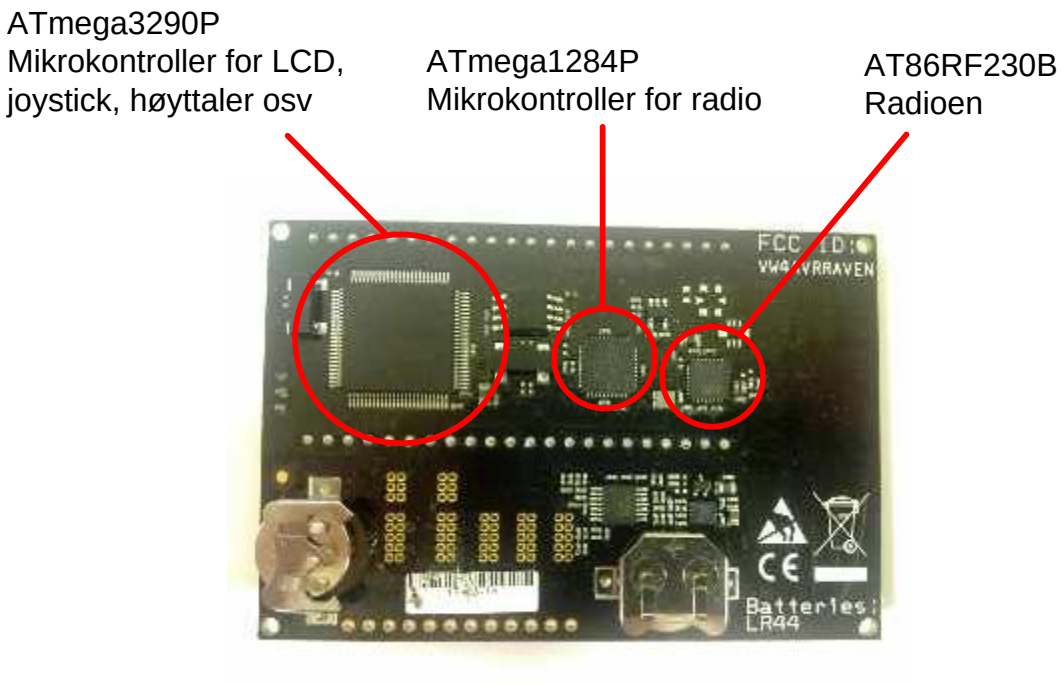
For å være i stand til å utvikle en egen implementering av WirelessHART-protokollen har vi nødvendigvis vært avhengige av diverse utstyr. Vi kom i kontakt med Atmel (se oppgavens innledning), og de har bidratt med alt av nødvendig utstyr, både maskinvare og programvare.
Maskinvaren vi fikk tildelt består av fire startpakker («RZ Raven Evaluation and Starter Kit»), to pakkesniffere («ATAVRRZ541 Packet Sniffer Kit») og to kombinerte debuggere og programmerere («JTAGICE mkII Debugger and Programmer»). Programvaren vi har brukt er «AVR Studio 4» til programmering av radioene via programmereren, og «Sensor Network Analyzer» for overvåking av kommunikasjonen mellom nodene i kombinasjon med pakkesnifferen. I tillegg har vi benyttet kildekodebiblioteket «AVR2025» som et grunnlag for utvikling av protokollen.
I dette kapitlet presenterer vi utstyret og programvaren, samt diskuterer noen erfaringer vi har gjort oss underveis. (Da leveransen fra Atmel ankom var vi svært spente på innholdet, og forventningen ble ikke lavere da vi så emballasjen. Se figurene Figur 5-2, Figur 5-3 og Figur 5-6 under).
De tre neste underkapitlene (kapitlene 5.1, 5.2 og 5.3) beskriver maskinvare og programvare som vi har benyttet uten nevneverdige endringer. Kapittel 5.1 beskriver selve brikken vi programmerer på, samt en gjennomgang av spesifikasjonene sammenlignet med WirelessHART-standardens krav, og er derfor nyttig lesing. De to etterførlgende kapitlene kan oppleves som oppramsende og til tider vel detaljerte, men vi mener likevel at de kan være interessante for noen lesere og har derfor valgt å ta dem med. Kapittel 5.4 beskriver biblioteket AVR2025, utgangspunktet for vår implementering, og vi har hentet både inspirasjon og allerede eksisterende løsninger fra dette biblioteket. For en dekkende forståelse av vår implementering mener vi derfor at kapittel 5.4 er viktig og relevant lesing.

5.1 Startpakken RZ Raven

Denne pakken inneholder det nødvendigste for et elementært trådløst nettverk og for å programmere dette. Innholdet i pakken er:

- To AVR Raven-kort med 2.4GHz radio, to mikrokontrollere og LCD-skjerm.
- En USB-pinne med 2.4GHz radio.
- Programvare for oppsett og analyse av nettverket.

Vi har ikke benyttet oss av USB-pinnen som fulgte kittet, da fungerende drivere ikke var tilgjengelige for våre Windows 7-maskiner. Vi gir derfor ingen videre beskrivelse av denne i teksten under.



Figur 5-1 Baksiden av et Raven-kort

Figur 5-1viser Raven-kortenes viktigste komponenter: to mikrokontrollere (ATmega3290P og ATmega1284P), samt en radio (AT86RF230B). ATmega3290P er den mikrokontrolleren som håndterer høyttaleren, LCD-skjermen, joystick og sensorene, mens ATmega1284P kontrollerer radioen. I vår implementering har vi kun forholdt oss til radioaktiviteter, og har derfor kun programmert mot den sistnevnte mikrokontrolleren. For å programmere radioen måtte vi benytte et ISP-grensesnitt (In-System Programming). Til dette grensesnittet måtte vi lodde på en medfølgende overgang som kunne kobles mot programmereren/debuggeren (JTAGICE-enheten).

Begge de nevnte mikrokontrollerne er knyttet til separate 32768 Hz klokkekrystaller, og på denne måten kan en applikasjon implementere en sanntidsklokke som holder rede på tiden, også når en node slår av radioen og mikrokontrolleren (går i dvalemodus).

Figur 5-2 Innholdet i RZ RAVEN-pakken

5.1.1 Radioen og fysiske krav

Som vi så i kapittel 4.2.4 stilles det krav til enhetens radio (basert på IEEE 802.15.4-2006 og WirelessHARTs modifikasjoner). I dette underkapitlet ser vi på hvorvidt vår radio oppfyller kravene.

Når det gjelder kravene rundt radioens skifte fra RX til TX og omvendt (som vi husker skulle skje innen 192 us), ser vi at radioen har en «Integrated TX/RX switch» [4], og siden radioen er bygget for å være kompatibel med IEEE 802.15.4 (som også krever skifte av retning innen 192 us på 2,4 GHz-båndet) må vi gå ut fra at kravet er oppnådd. Det blir også nevnt at det er ønskelig å kunne skru på radioen på under 4 ms, noe vår radio oppnår på under 1 ms.

Overføringseffekten skal på WirelessHART-enheter være 10 dBm ±3 dB (som tilsvarer mellom 5 og 20 dBm), og vi kan se av spesifikasjonen at radioen vi jobber med har et effektområde på -17 dBm til 3 dBm. Her når altså ikke vår radio opp til kravene som stilles av WirelessHART, men vårt fokus er ikke på effekt og energibruk så vi kan se bort fra dette i vårt arbeid. De funksjonelle egenskapene til vår implementering vil ikke påvirkes av dette. I en situasjon med driftsetting av WirelessHART-protokollen vil det nok være nødvendig å se

etter en radio med mer effekt, eller eventuelt legge på en forsterker (LNA – Low-noise Amplifier) for å oppnå nødvendig effekt. Maskinvarebeskrivelsen av radioen AT86RF230B påpeker for øvrig at den er ment å kunne håndtere mange protokollene, deriblant WirelessHART [4].
Kravet til sensitivitet forstod vi som -85 dBm eller bedre (kapittel 4.2.4), og da sensitiviteten på radioen oppgis å være -101 dBm er dette kravet oppnådd.

5.2 Programmereren og debuggeren JTAGICE mkII

Figur 5-3 JTAGICE mkII debugger

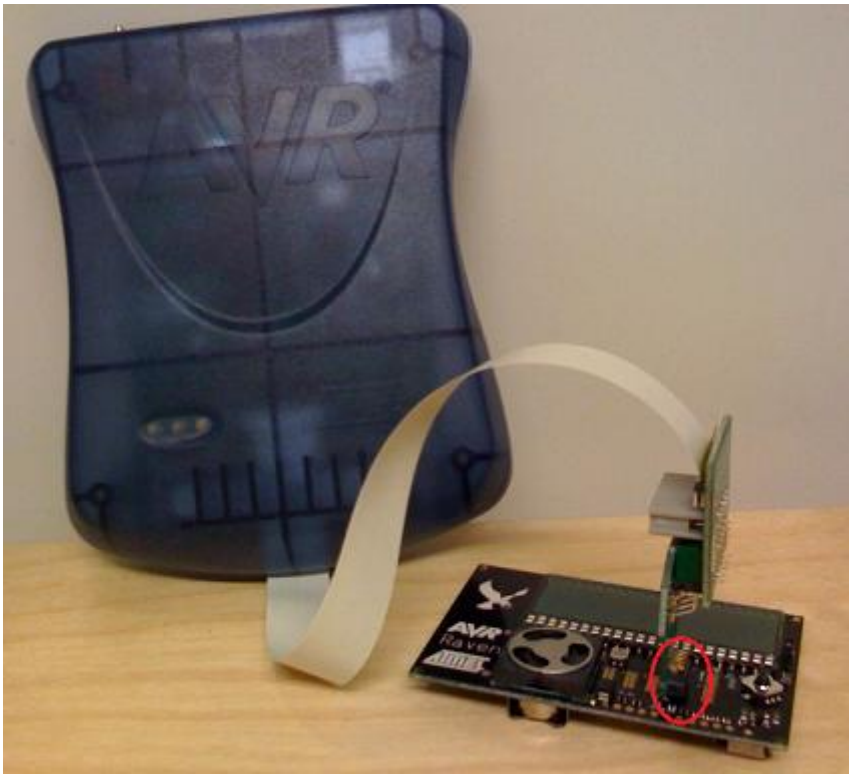
5.2.1 AVR Studio 4

AVR Studio 4 er Atmels egenutviklede IDE (Integrated Development Environment) for programmering på alle programmerbare AVR-enheter som finnes på markedet. (AVR er en «familie» mikrokontrollere utviklet av Atmel). Her har programmereren god oversikt over prosjektet og det er muligheter for emulering av AVR-enheter og debugging direkte på kortet.

For at AVR Studio 4 skal oppdage riktig utstyr er det anbefalt først å slå på AVR-enheten (RAVEN noden), så slå på JTAGICE enheten og til slutt starte AVR Studio 4 [42]. Det gjøres også oppmerksom på at alt annet AVR-utstyr burde være koblet fra PC-en for å unngå at feil utstyr oppdages. Vi har til tider opplevd at AVR Studio 4 ikke oppfører seg som forventet og omstart av programmet har vært nødvendig, men ser man bort fra noen av programmets små feil, og heller merker seg interaksjonsmulighetene mot Raven-kortet (og alle medlemmene av AVR-familien for øvrig) som tilbys er AVR Studio 4 et kjærkomment hjelpemiddel.
Det finnes også en plug-in til Eclipse for programmering mot AVR-utstyr om man skulle foretrekke det [43].

On-chip debugging

Alle AVR-enheter med et JTAG-grensesnitt har en såkalt «On-chip debug»–logikk. Via dette grensesnittet har man tilgang til mikrokontrollerens interne resurser, og man gis muligheten til å debugge (og programmere) rett på enheten. Dette er forskjellig fra tradisjonelle debuggingteknikker hvor man emulerer oppførselen til utstyret så nære utstyret som mulig. Figur 5-4 viser en programmerer/debugger koblet til et Raven-kort via et JTAG-grensesnitt (ringet ut i rødt).

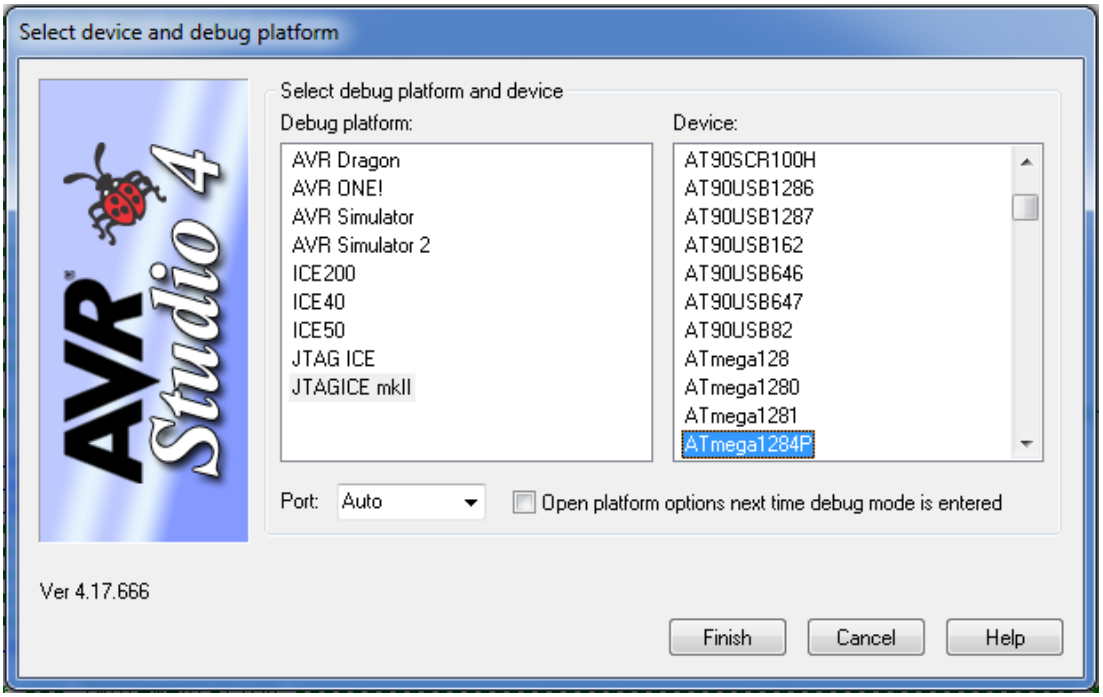


Figur 5-4 JTAG-grensesnitt som gir tilgang til mikrokontrollerens interne resurser

Programmering via JTAGICE mkII

Så sant overgangen til ISP-grensesnittet er loddet ordentlig på kortet man skal programmere (i vårt tilfelle Raven-kortet) og programmereren/debuggeren er koblet til PC-en og kortet, er det bare å laste inn prosjektet i IDE-en så er man i utgangspunktet klar. Vi benytter oss av biblioteket AVR2025 (se kapittel 5.4) som også inneholder kompilerbare prosjekter man kan prøve ut – uten å endre kildekoden.

Når man i AVR Studio 4 oppretter et prosjekt, eller åpner et prosjekt for første gang, velger man hvilket utstyr det skal programmeres mot (Figur 5-5).



Figur 5-5 Dialog i AVR Studio 4 for spesifisering av utstyr

5.3 Pakkesnifferen ATAVRRZ541

Underveis i implementeringen har vi hatt behov for å visualisere den faktiske pakkestrukturen og -flyten som sendes fra en node til en annen. Til dette har vi benyttet sniffer-settet ATAVRRZ541. Settet inneholder brikken STK541, en 2,4 GHz programmerbar radio, samt programvaren «Sensor Network Analyzer» for visualisering av IEEE 802.15.4 datatrafikk som oppdages av snifferen.

Figur 5-6 RZ541 pakkesniffer

5.3.1 STK541 og programmerbar radio

STK541 er et tilkoblingsbrett med USB-tilkobling mot PC. Dette kan benyttes i flere AVR -utviklingsscenarioer, og i vårt tilfelle benyttes det til å danne en pakkesniffer sammen med radioen som følger med.

Den programmerbare transceiveren som kobles til STK541-brettet er av samme type som radioen på Raven-brettene (AT86RF230B). Forskjellen er at den her er koblet til en utvidet antenne og at den kontrolleres av en annen mikrokontroller (ATmega1281v).

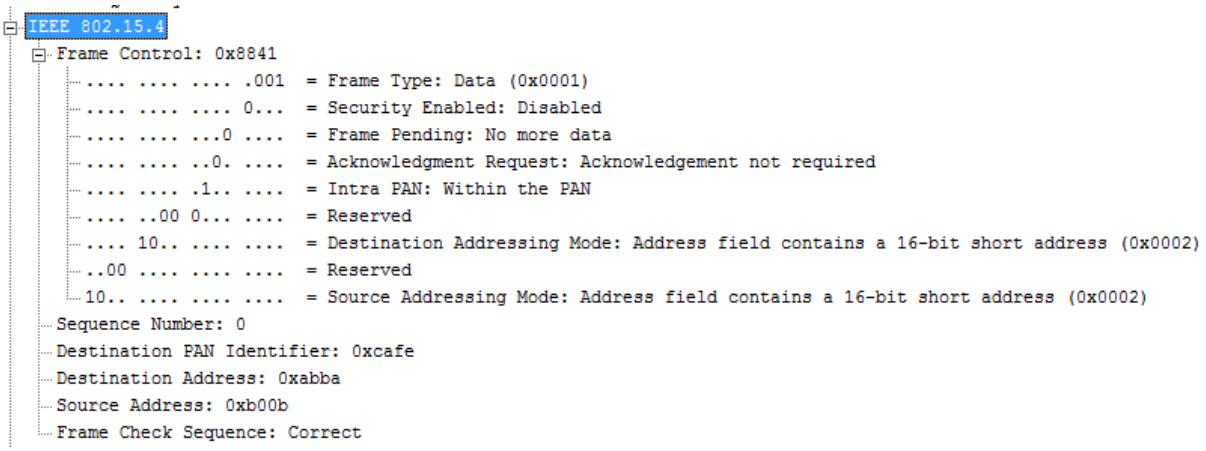
STK541-brettet kan altså benyttes i en rekke forskjellige situasjoner med annet Atmel-utstyr, og det kommer ikke forhåndsprogrammert til vårt formål. For å kunne fungere som en sniffer må man programmere inn sniffer-firmware på STK541-brettet [44]. Dette gjøres i AVR Studio 4 via JTAGICE.

5.3.2 Sensor Network Analyzer

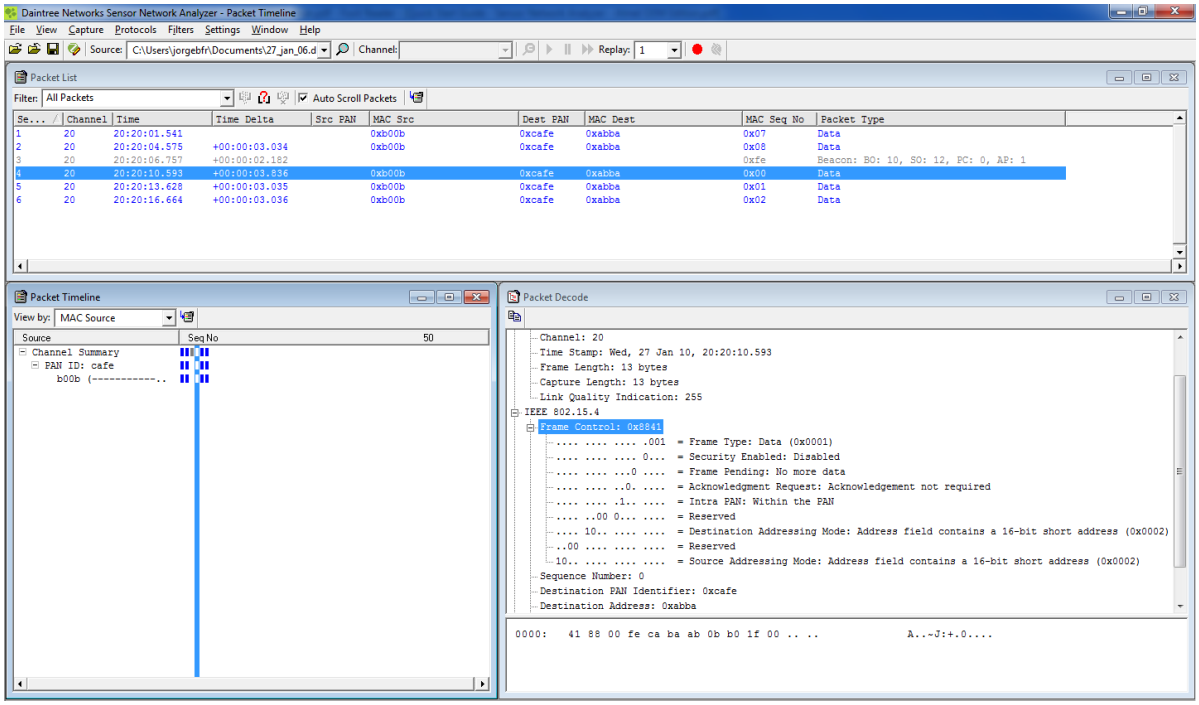
For å kontrollere snifferen benytter vi den medfølgende programvaren «Sensor Network Analyzer (SNA) – Basic edition» fra Daintree Networks. Vi har benyttet oss av versjon 2.1.0.19 som ved oppstart var den nyeste versjonen tilgjengelig.

Med SNA kan vi både kontrollere snifferen og visualisere IEEE 802.15.4-trafikk.

Visualiseringen fremstår som veldig oversiktlig og hensiktsmessig. Man har et vindu med generell pakkeinformasjon (sekvensnummer, tidspunkt, tid siden forrige pakke, adresser, pakketype m.m.), et vindu som viser en tidslinje over datatrafikken på den valgte kanalen og et vindu hvor man kan inspisere hver enkelt pakke på bit-nivå. Figur 5-8 viser en skjermdump fra SNA hvor vi har overvåket datatrafikken på et stadium av vår egen implementering hvor vi nylig har innført WirelessHARTs pakkestruktur. SNA er utviklet for å tolke IEEE 802.15.4-trafikk, og da spesielt ZigBee, men om vi inspiserer en WirelessHART-datapakke nærmere (Figur 5-7) ser vi at kontrollfeltet (sammen med den ledende 41_{16} som vi i kapittel 4.3.4 så ble lagt til alle WirelessHART-pakker) matcher starten på en 802.15.4 -pakke og SNA kan gi oss fornuftig informasjon som for eksempel adresselengder.



Figur 5-7 Fremvisning i SNA av kontrollfeltet til en WirelessHART datapakke



Figur 5-8 Sensor Network Analyzer

5.4 AVR2025

Nå har vi gått gjennom utstyret vi har implementert på, og det er på tide å se nærmere på grunnlaget for selve implementeringen.

Da vi skulle programmere WirelessHART-protokollen undersøkte vi tilgjengeligheten på en IEEE 802.15.4-implementering som vi kunne ta utgangspunkt i. Siden utstyret kommer fra Atmel var det naturlig å ta utgangspunkt i et bibliotek som var tilpasset dette utstyret, og kodebiblioteket AVR2025 tilfredsstiller nettopp dette. I dette kapittelet vil vi legge frem de viktigste punktene rundt biblioteket vi benytter oss av i vår implementering.

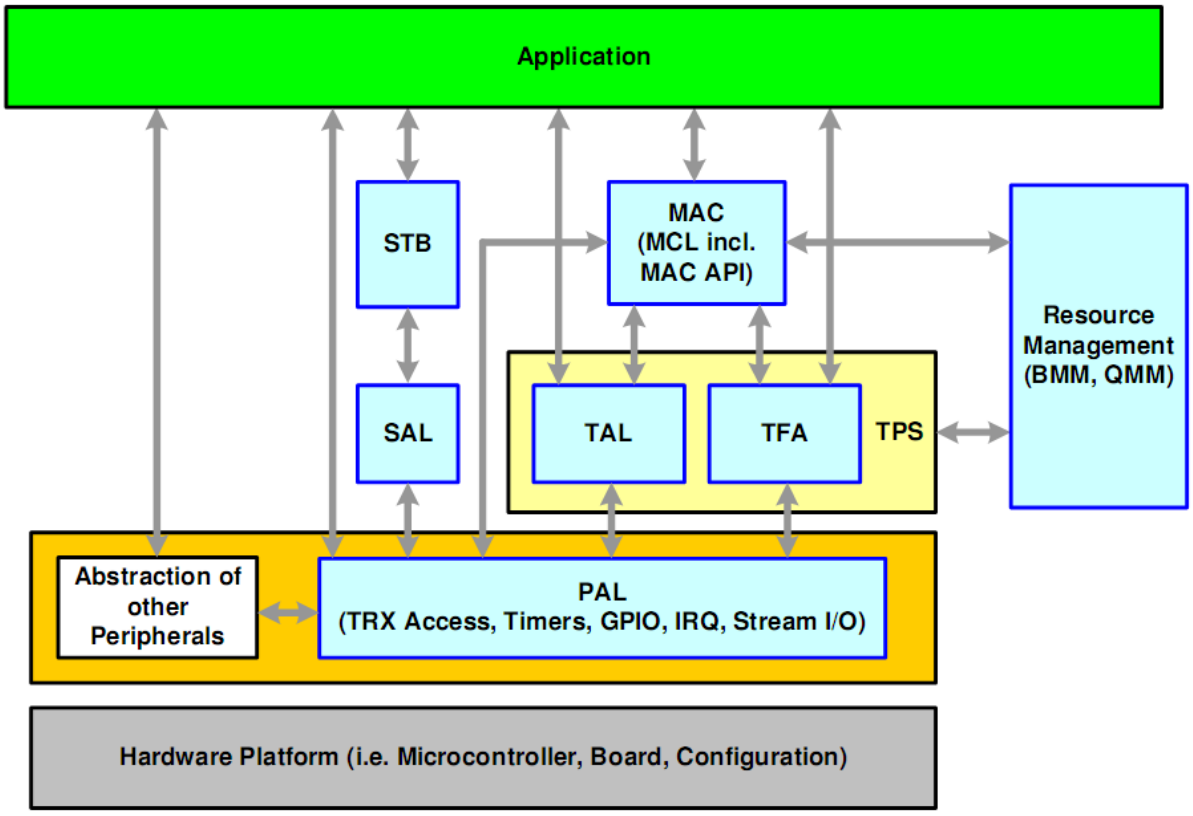
Atmel tilbyr store mengder ressurser for utvikling og videreutvikling av løsninger tilpasset deres maskinvare. Ressursene som tilbys er hovedsaklig brukerveiledninger, programvare og dataark, og er tilgjengelig på Atmels hjemmesider [45].

5.4.1 Generelt

AVR2025 er et kodebibliotek som tilbyr en fullverdig IEEE 802.15.4-2006-implementering. Det er bygget opp for å støtte alle Atmels mikrokontrollere, og det er i tillegg lagt ved en del eksempelapplikasjoner på toppen. Oppbyggingen er relativt modulær, i det formål å gjøre det enkelt å bytte ut de delene som er nødvendig for de ulike mikrokontrollerene og

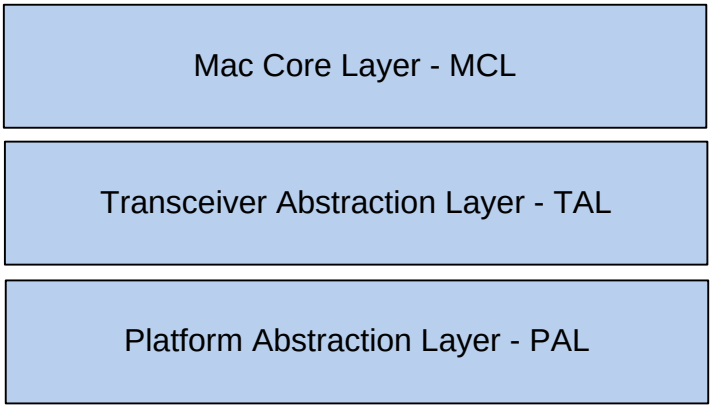
applikasjonene fra Atmel, uten å måtte skrive alt på nytt. Samtidig er det mulig å definere hvilket lag man vil benytte som det øverste laget fra biblioteket mot sin egen applikasjon. Figur 5-9 er hentet fra bibliotekets dokumentasjon [46], og viser den modulære oppbyggingen av biblioteket.

De viktigste modulene er Platform Abstraction Layer (PAL), Transceiver Abstraction Layer (TAL) og MAC Core Layer (MCL) (se Figur 5-10). Alle tre lagene har funksjoner for initiering og oppgavehåndtering. Oppgavehåndteringen kalles fra applikasjonen, som fører til at de korresponderende oppgavehåndteringsprosedyrene i MCL, TAL og PAL blir kalt. Om



Figur 5-9 MAC-lagets arkitektur

applikasjonen sørger for at dette skjer hyppig nok opprettholdes hvert lags tilstandsmaskiner og deres oppgavekøer blir prosessert. Henter vi ut disse tre lagene fra Figur 5-9 blir det enklere se hvordan de forholder seg til hverandre. Dette er illustrert i Figur 5-10.



Figur 5-10 Logisk skille mellom PAL, TAL og MCL

5.4.2 PAL – Platform Abstraction Layer

Vi skal nå gå gjennom de tre lagene i Figur 5-10, og starter med PAL og jobber oss oppover.

PAL-modulen inneholder (som navnet tilsier) den funksjonaliteten som ligger nærmest mikrokontrolleren og maskinvareplattformen som benyttes. Valg av mikrokontroller avgjør dermed hvilke deler av biblioteket man skal benytte seg av. PAL tilbyr grensesnitt direkte mot alle andre moduler i modellen (med unntak av ressurshåndteringen) som vist ved pilene i Figur 5-9. På denne måten blir biblioteket plattformuavhengig og enkel å migrere til annen maskinvare, samt at det åpner for å utelate bibliotekets TAL og MCL og heller skrive sine egne implementeringer.

I PAL har utvikleren en tilnærmet direkte tilgang til radioens registre og minne. Radiospesifikk funksjonalitet er, som vi skal se under, videre abstrahert i TAL.

5.4.3 TAL – Transceiver Abstraction Layer

TAL-modulen inneholder funksjonalitet som muliggjør IEEE 802.15.4 MAC-støtte, og som er spesifikk for radioen man programmerer på. TAL benytter seg utelukkende av tjenestene som PAL tilbyr for å kommunisere med radioen.

For TAL, som for PAL, finnes det flere implementeringer i biblioteket, og valg av implementering er avhengig av hvilken radio man jobber med. I hver av disse tilgjengelige implementeringene er det bygget inn programkode som utnytter radioens spesifikasjoner. Blant annet er det mulig å utføre ACK-funksjonalitet i selve radioen ved at den plattformspesifikke modulen programmerer det aktuelle registeret. Siden ACK-ing fungerer nokså annerledes i WirelessHART enn i standard IEEE 802.15.4, kan vi ikke benytte oss av

denne automatiske ACK-funksjonaliteten i vår implementering. Det samme gjelder for enkelte andre innebygde funksjoner i radioen. Vi vil komme tilbake til dette i kapittel 6.
For øvrig inneholder TAL-modulen utsending og mottaking av rammer, TAL tilstandsmaskin, PAN informasjon som angår TAL (typisk PIB-attributter omtalt i kapitlene 3.2.2, 4.2.2 og 6.3.2), håndtering av CSMA, energideteksjon, strømhåndtering, avbruddshåndtering knyttet til radioen og initiering/tilbakeføring av TAL-tilstander.

5.4.4 MCL – MAC Core Layer

I MCL-modulen har Atmel samlet de delene av IEEE 802.15.4-standarden som lagene over benytter seg av, i form av et grensesnitt opp mot høyere lag. Foruten selve MAC-grensesnittet inneholder MCL MAC-moduler som håndterer videresending av pakker, datapakketjenester, nettverksrelaterte administrasjonstjenester, beaconhåndtering, prosessering av innkommende pakker og lagring av PAN-informasjon som angår MAC-laget.
Ved utsending av pakker hentes pakken som står for tur ut av MAC-lagets kø. og sendes deretter til TAL. Om pakken er på vei ut (i motsetning til om pakken er en administrasjonspakke ment for denne nodens datalinklag) vil en bekreftende funksjon (som utsteder en confirmation tjenesteprimitiv) bli kalt for å gi applikasjonen beskjed om sending av pakke, og pakken vil bli sendt via en TAL-funksjon.
MAC-modulen for å håndtere innkommende pakker leverer disse til applikasjonen ved å legge dem i kø og kalle på en funksjon som indikerer ovenfor det høyere laget at en innkommet pakke er klar (indication tjenesteprimitiv).
Biblioteket tilbyr også en modul for beaconhåndtering som sørger for opprettelse, forsendelse og synkronisering av beacons. Dette er dog en modul vi kan se helt bort ifra i vår WirelessHART-implementering, siden WirelessHART ikke benytter seg av beacons.
Modulen for lagring av PAN informasjon som angår MAC-laget benyttes av modulene for både datapakketjenester, nettverkstjenester og beaconhåndtering. I denne modulen lagres informasjon relatert til nettverket, for eksempel MAC-adresser, maksimalt antall back-offs før man gir opp osv. Denne modulen kalles PIB (Personal Area Network Information Base), etter inspirasjon fra en tilsvarende struktur som defineres av IEEE 802.15.4-standarden, men da for det fysiske laget (mer om dette i kapittel 6.3.2).

5.4.5 Andre moduler i biblioteket

Ut over de modulene vi nettopp har gått gjennom, inneholder biblioteket også andre moduler. Modulen for ressurshåndteringen er av en slik art at den lar seg gjenbruke i vår implementering. Ressurshåndteringsmodulen inneholder funksjonalitet for allokering og frigjøring av minne, samt organisering og oppdatering av køer. For eksempel hendelseskøer og køer for innkommende datapakker.

Det finnes moduler som ikke blir benyttet i vår implementering, men som senere kan bli nyttige. Blant annet en sikkerhetsmodul (Security Abstraction Layer – SAL) og et grensesnitt til dette (Security Toolbox – STB) som sammen virker som et fornuftig utgangspunkt når sikkerhetsmekanismene i WirelessHART skal implementeres.

Til slutt har Atmel også trukket ut egenskaper som deres radioer kan tilby utover det som kreves i IEEE 802.15.4 under navnet Tranceiver Feature Access (TFA).

5.4.6 Strømsparing

Siden det er interessant med lengst mulig levetid på de batteridrevne sensornodene er effektiv bruk av strøm avgjørende. I AVR2025 ligger det strømsparingsfunksjoner i MAC-laget.

IEEE 802.15.4 og WirelessHART har forskjellige behov for å ha radioen i dvalemodus, men basisfunksjonene rundt energibruk i AVR2025 er oversiktlige og lar seg greit benytte og konvertere til en WirelessHART-implementering.

Det er mulig å slå av og på radioen via funksjoner i MAC-laget, eller mer direkte i TAL. Atmel understreker dog at bruk av strømsparingsfunksjoner i TAL fra andre lag enn TAL vil kunne føre til uviss oppførsel [46].

5.4.7 Valg av applikasjon

AVR2025 kommer med et utvalg eksempelapplikasjoner som kan fungere enten som inspirasjon eller også som utgangspunkt for modifisering og utvidelse. Til vår implementering av WirelessHART bestemte vi oss for applikasjonen med navnet «App_1_Nobeacon». Dette er en applikasjon som kun implementerer det mest grunnleggende for et IEEE 802.15.4-nettverk. Applikasjonen tar utgangspunkt i MAC-laget som det høyeste laget.

Eksempelapplikasjonen definerer to noder. En PAN-koordinator og en vanlig node (device) som enten er konfigurert med full eller redusert funksjonalitet. (Se kapittel 3.2.1 om komponenter for mer om noders funksjonelle modi). I WirelessHART vil alle nodene fungere som fullt funksjonelle noder, så vi tok utgangspunkt i den vanlige noden fra applikasjonen

konfigurert med full funksjonalitet. Denne, som mange av eksemplene i AVR2025, er laget som en ZigBee-applikasjon og det måtte nødvendigvis gjøres endringer for at den skulle passe til vårt formål. I neste kapittel ser vi på vår implementering.

6 Implementering

I dette kapitlet går vi gjennom hvordan vi har valg å implementere deler av datalinklaget for WirelessHART, samt hvilke tilpasninger vi har gjort i det fysiske laget i kodebiblioteket fra Atmel (AVR2025). De aller fleste bestanddelene er implementert slik vi har lest og forstått standarden, men som vi har vært inne på tidligere er det mange deler av spesifikasjonen hvor utvikleren står fritt til å velge løsning så lenge det overordnede kravet tilfredstilles og interoperabilitet er sikret. Vårt fokus har vært å få de ulike programvarekomponentene til å fungere etter spesifikasjonen, og optimaliseringer av de ulike algoritmene har vært sekundært. Det vil derfor sannsynligvis være et potensiale for forbedringer av de foreslåtte algoritmer og implementeringer i dette kapittelet, både med tanke på tids- og energiforbruk. Vi mener imidlertid at de fungerer som et godt utgangspunkt for videre arbeider med en fullstendig implementering av WirelessHART-protokollen. De delene av implementeringen som er tilnærmet fullstendige er testet i neste kapittel, og vi vil der se at vi klarer å holde oss godt innenfor spesifikasjonens krav på viktige områder.

Før vi startet med vår egen programmeringen kartla vi tidligere arbeider som også tok for seg implementering av WirelessHART. Før vi går inn på vårt eget arbeid skal vi oppsummere de viktigste arbeidene vi fant i den forbindelse

.

6.1 Tidligere arbeider

Det finnes foreløpig få implementeringer av WirelessHART hvor prosessen underveis er dokumentert. Kungliga Tekniska högskolan i Stockholm (KTH) har et visst miljø på sin avdeling for Electrical Engineering, og herfra kommer blant annet en masteroppgave som tar for seg en implementering på en fysisk brikke [47], samt en masteroppgave som omhandler implementering av en WirelessHART-simulator på toppen av TrueTime-simulatoren [48]. Sistnevnte danner også basis for en rapport publisert gjennom IEEE [31]. I tillegg finnes en kort rapport av en implementering utført av en gruppe utviklere fra Hartcomm, Emerson og University of Texas [49], samt en mer teoretisk studie av implementasjonstekniske spørsmål rundt WirelessHART-standarden utført av forskere ved blant annet ABB, Sintef og NTNU [38]. Vi har gått gjennom disse arbeidene for å se om vi kunne hente erfaringer og lærdom av dem, og en kort beskrivelse av hver av dem er gitt i de påfølgende avsnitt. Den generelle oppfatningen er at tekstene ikke er direkte relevante for oss på dette stadiet, men de ga oss likevel innsikt og tolkninger som var nyttige som bakgrunnskunnskap.

«WirelessHART – Implementation and Evaluation on Wireless Sensors» [47]

Dette er en masteroppgave skrevet av David Gustafsson ved KTH i 2009. Oppgaven tar for seg implementeringen av WirelessHART på toppen av et Contiki operativsystem, samt kjøring og evaluering av dette på TMote Sky-noder fra Moteiv. Contiki er et åpen kildekode-operativsystem skrevet spesielt for innebygde systemer som krever nettverk, og var, som vi skal komme tilbake til, en aktuell kandidat for grunnlaget for vår implementering. Siden Gustafsson har valgt å fokusere på nettverks- og transportlaget, og ikke har implementert datalinklaget, er det begrenset hvor mye av denne oppgaven som er relevant for oss. Vårt fokus har nesten utelukkende vært på det fysiske laget og datalinklaget. Vi kan imidlertid lese at Gustafsson mener å vise at WirelessHART lar seg implementere og kjøre på noder med begrensede ressurser, og vi får i tillegg et innblikk i en implementering som er basert på et underliggende operativsystem.

«Implementation of a WirelessHART simulator and its use in studying packet loss compensation in networked control» [48]

Også dette er en masteroppgave fra KTH, skrevet av Mauro De Biasi i 2008. Dette er ikke en implementering på en fysisk node, men derimot på den MatLab-baserte simulatoren TrueTime i den hensikt å bygge en WirelessHART-simulator. Oppgaven fokuserer hovedsaklig på bruken av simulatoren i studier av pakketap og resultatene dette gir, men har også dedikert et par sider til det implementeringsspesifikke. Noe av det er hentet mer eller mindre direkte fra spesifikasjonen og er derfor ikke så interessant for oss, men De Biasi presenterer i tillegg en algoritme for håndtering av planlagte tidslommer som virker fornuftig. Simulatoren er imidlertid ikke laget for å undersøke energiforbruk, og dette er dermed ikke tatt hensyn til i algoritmen. De Biasi benytter en algoritme hvor det for hver enkelt tidslomme blir kjørt en del prosessering for å vurdere om det skal foregå linkbehandling eller ikke. Vi anser dette som lite energieffektivt, da alle linker (og i tilfelle med sendelinker alle ventende pakker) må itereres for hver eneste tidslomme, det vil si med 10 millisekunders mellomrom, med en slik strategi. Vi har i stedet valgt å bare iverksette linkbehandleren der det faktisk skal skje behandling. Dette medfører en noe mer krevende linkfordeler, men denne skal til gjengjeld bare kjøres ved enkelte gitte hendelser. Denne strategien fører derfor til at algoritmen beskrevet i De Biasis oppgave ikke lar seg direkte overføre til vår. I stedet fungerer den fint som en oppsummering av linkbehandlingsprossessen, og er benyttet som bakgrunnsinnformasjon i vår egen utvikling.

«WirelessHART: Applying Wireless Technology in Real-Time Industrial Process Control» [49]

Denne teksten er skrevet av syv forskere og utviklere fra Hartcomm, Emerson og University of Texas, og publisert gjennom IEEE i 2008. Den omhandler implementeringen av en WirelessHART-protoyp og bruken av denne i et enkelt demonstrasjonsnettverk. Fokus ligger på de utfordringer de møtte underveis i arbeidet, og foreslåtte løsninger på disse. Noen av utfordringene er ikke relevante for oss på grunn av det noe snevrere omfanget vi har valgt, men timere og tidsavbrudd, synkronisering og tilstandsmaskindesign kommer vi ikke utenom. I motsetning til forfatterne har vi imidlertid allerede fungerende mekanismer for timere og tidsavbrudd gjennom Atmels kodebibliotek AVR2025, og problemstillingene som tas opp i den forbindelse er derfor mindre relevante for oss. Når det gjelder synkronisering og designet av tilstandsmaskinen bidrar teksten til å utheve og klargjøre enkelte detaljer fra spesifikasjonen, og den presenterer et oversiktlig forslag til arkitektur av datalinklaget som vi har hatt god nytte av i vårt eget arbeid. Vi presenterer i kapittel 6.4 vår datalinklagsarkitektur, som i stor grad er basert på forslaget i denne artikkelen.

«When HART goes Wireless: Understanding and Implementing the WirelessHART Standard» [38]

I motsetning til de andre tekstene tar ikke denne for seg en konkret implementering, men ser i stedet på protokollen fra et mer teoretisk perspektiv. Teksten er skrevet av fire forskere fra ABB, Q2S, NTNU og Sintef, og publisert gjennom IEEE i 2008. Formålet her er å identifisere eksisterende metoder og algoritmer som er spesielt egnet til en implementering av WirelessHART, og da spesielt på MAC-laget og hos nettverksadministratoren. Teksten inneholder mange interessante betraktninger, og flere algoritmer trekkes frem og diskuteres, blant annet for ruting, linkplanlegging og tidssynkronisering. Disse algoritmene er imidlertid hentet fra generell nettverksteori, og ikke skrevet spesielt for WirelessHART. De må derfor i større eller mindre grad tilpasses WirelessHART-spesifikasjonen, og teksten diskuterer ikke inngående hvordan dette skal gjøres. Da algoritmene heller ikke alltid er veldig trivielle, ser dette ut til å være et omfattende arbeid. Vår oppfatning er at denne teksten kan være en uvurderlig ressurs i et langsiktig arbeid med å implementere en god og fullverdig, energieffektiv WirelessHART-protokoll. Samtidig ser vi, som nevnt innledningsvis i kapitlet, et behov for å holde kompleksiteten nede i startfasen og heller konsentrere oss om å få

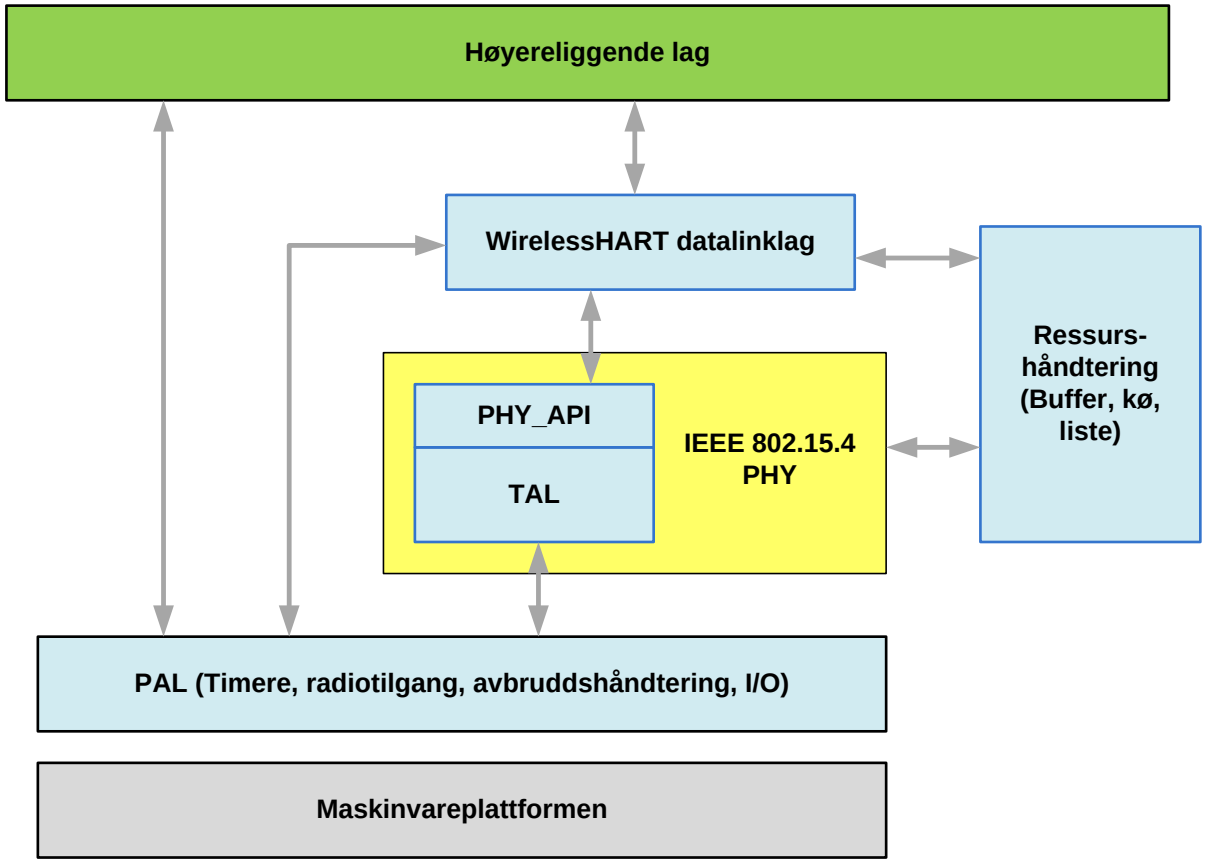
funksjonaliteten på plass. Av den grunn har vi valgt ikke å fokusere på de algoritmer og løsninger som gjennomgås i teksten i denne omgang.

6.2 Arkitektur

Et viktig spørsmål ved implementeringen var om vi skulle basere den på et operativsystem eller ikke. I avsnittene over så vi et eksempel på en delvis implementering av WirelessHART på Contiki OS, og dette er et operativsystem som også lar seg kjøre på Raven-brikkene (se for eksempel [50]) og som leveres med en IEEE 802.15.4-implementering. Fordelen med et operativsystem er innebygd støtte for prosesser, tråder og multitasking, og i så måte kunne nok Contiki vært til god hjelp. Imidlertid finnes den samme nettverksfunksjonaliteten i Atmels kodebibliotek AVR2025, som i hovedsak består av en IEEE 802.15.4-2006-implementering, og som i tillegg legger til rette for en enkel, timer-drevet planlegger. WirelessHART er en deterministisk og lineær protokoll der alle hendelser innad i protokollen skjer etter hverandre på predefinerte tidspunkt, og behovet for å kjøre flere prosesser eller tråder parallelt er derfor ikke til stede. Det er rimelig å anta at en implementering direkte mot maskinvaren, slik AVR2025-biblioteket er, vil gi lavere minne- og energiforbruk enn et operativsystem. Siden AVR2025-biblioteket i tillegg kom prekonfigurert for bruk med maskinvaren (Raven-settet) sammen med IDE-et AVR Studio 4 falt valget på denne løsningen.

Valget av AVR2025 som grunnlag for implementeringen la naturlig nok en del føringer for arkitekturen. Vi har allerede sett hvordan AVR2025 er inndelt i ulike moduler, herunder MCL, PAL og TAL (Figur 5-9 og Figur 5-10) som de mest sentrale. Hva av dette som utgjør det fysiske laget og hva som utgjør datalinklaget er ikke helt opplagt, blant annet fordi enkelte MAC-funksjoner er trukket ned i maskinvaren i radioen og dermed initieres fra det som normalt ville utgjøre PHY-programvaren. Automatisk retransmisjon, CSMA/CA og automatisk ACK-ing er eksempler på dette. Det er ingenting i WirelessHART som tilsier at vi ikke kan holde oss til en slik krysslagsdesign, men siden datalinklaget i WirelessHART er såpass ulikt IEEE 802.15.4 MAC har vi valgt å fjerne alt som minner om MAC-funksjoner fra TAL-modulen, samtidig som vi legger et grensesnitt på toppen, `phy_api`, som implementerer alle tjenesteprimitivene ned mot det fysiske laget (Figur 6-1). På den måten kan TAL sammen med `phy_api` i grove trekk sies å utgjøre det fysiske laget. PAL inneholder en del funksjonalitet som blant annet har med skriving til registere og tidshåndtering å gjøre, og går derfor på tvers av lagene. MCL-laget fra kodebiblioteket

AVR2025 er for ulikt datalinklaget i WirelessHART til å kunne benyttes, og vi har derfor erstattet det med et eget WirelessHART-datalinklag. For øvrig har vi beholdt buffer- og køhåndteringen i AVR2025, samt utvidet med en listehåndterer for situasjoner der en liste er bedre egnet enn kø (omtalt senere i kapitlet). Figur 6-1 viser den endelige, modifiserte AVR2025-arkitekturen. Den opprinnelige arkitekturen er illustrert i Figur 5-9 på side 84.
Datalinklaget har i seg selv en logisk todeling: En del som består av TDMA tilstandsmaskinen og dens linkplanlegger og sende- og mottaksmotorer, og en del som består av en rekke kommunikasjonstabeller og MAC-PIB-en (PAN Information Base). Disse delene korresponderer med henholdsvis MAC-laget og LLC-laget som omtalt i kapittel 4.3. Vi skal se nærmere på den interne arkitekturen i datalinklaget i kapittel 6.4.



Figur 6-1 Modifisert AVR2025-arikitektur

Siden hendelser i WirelessHART foregår sekvensielt og på predefinerte tidspunkt endres programflyten i protokollen i forhold til en standard IEEE 802.15.4-implementering. I en komplett IEEE 802.15.4-implementering må man ta hensyn til at pakker skal kunne sendes eller mottas mer eller mindre når som helst, og derfor er det viktig å kunne køe hendelser hvis

det skjer mye på kort tid. Av denne grunn er alle tjenesteprimitivene i AVR2025 opprinnelig implementert som datastrukturer som legges på en hendelseskø, og deretter behandles ved neste anledning. Vi valgte å gå vekk fra denne løsningen siden TDMA-protokollen i WirelessHART sørger for at det aldri foregår mer enn én hendelse om gangen, og i stedet stiller strenge krav til tidsstyring. (Unntaket her er data fra applikasjonen, f.eks. en sensorapplikasjon. Dette må håndteres på en asymmetrisk måte i de høyereliggende lagene). Siden all sending og mottak av data skal skje på bestemte tidspunkter, lar dette seg dårlig forene med prinsippet om køing og behandlig ved neste anledning. Derfor har vi implementert alle tjenesteprimitivene som rene funksjonskall med tilhørende parametere, i motsetning til datastrukturer som køes. En tjeneste vil dermed bli utført umiddelbart ved forespørsel. I teksten har vi referet til implementeringen av en tjenesteprimitiv med kildekodeformatering (eks. `transmit_request(..)`) og selve tjenesteprimitivet med formateringen som benyttes i standarden (eks. TRANSMIT.request). Dette fungerer som en distinksjon mellom spesifikasjonen av en tjenesteprimitiv og vår implementering av den.

De hendelser som skal utføres på spesifikke tidspunkter, slik som den faktiske sendingen av en pakke, håndteres av timer-funksjonaliteten i PAL-modulen i AVR2025-biblioteket. Måten dette gjøres på er å sende med en peker til funksjonen som siden skal utføre hendelsen sammen med tidspunktet funksjonen skal eksekveres til denne tidshåndtereren. Med andre ord en såkalt *callback*-strategi. Tidshåndtereren i AVR2025 er implementert i programvaren, og kjører dermed i samme prosess som resten av protokollen. Derfor er den avhengig av å bli kjørt så ofte som mulig for å kunne utføre funksjonskallene til rett tid, og dette er ganske enkelt løst ved å gjøre et kall på tidshåndtereren inne i en evig while-løkke. Hver gang tidshåndtereren oppdager at et funksjonskall skal gjøres vil den utføre dette funksjonskallet, og så returnere til den evige løkken igjen. I pseudokoden under har vi forsøkt å vise hvordan dette i realiteten driver hele protokollen vår.

```
* Utfør initiering av noden (nullstill klokke, last inn PIB osv)

* Planlegg første link som skal håndteres (lytte etter advertise etter X
  antall sekunder) og send til tidshåndtereren

while(true) {
  if (tiden er ute for ventende funksjon) {
      * Utfør ventende funksjon
  }
}
```

En ulempe med denne «pollingen» som pseudokoden over viser, er at det vanskeliggjør soving av mikrokontrolleren, noe vi kommenterer i evalueringen i kapittel 7.5

I vår implementering blir ikke flere enn ett ventende funksjonskall køet om gangen, siden vi i WirelessHART sjelden har oversikt over mer enn ett steg frem. Dette skyldes at utførelsen av en hendelse kan påvirke hva som blir den neste hendelsen (eksempelvis kan feil i overførsel føre til at retransmisjon blir neste steg. Dette avgjøres av linkplanleggeren og beskrives nærmere senere). Derfor må det ventende kallet alltid lede til at et nytt funksjonskall legges på vent, for å unngå at vi blir fast i den evig løkken uten noe mer å utføre. Vi skal senere i kapittelet se hvordan linkplanleggeren står sentralt i dette.

6.3 Det fysiske laget

Selv om WirelessHART benytter seg av det fysiske laget til IEEE 802.15.4-2006, som også AVR2025-biblioteket implementerer, må visse modifikasjoner av AVR2025-bibliotekets fysiske lag gjøres for å tilpasse det til WirelessHART. For det første har ikke AVR2025 noe klart skille mellom det fysiske laget og datalinklaget, siden flere av datalinklagets funksjoner som ACK-ing og automatisk retransmisjon, som tidligere nevnt, er implementert i maskinvaren. Disse maskinvareimplementeringene er tilpasset et IEEE 802.15.4-datalinklag, og således ikke kompatible med WirelessHART, og må derfor kobles ut.

I tillegg må PIB-strukturen (PAN Information Base) tilpasses WirelessHART i henhold til beskrivelsen i kapittel 4.2.2, siden en del av de valgfrie parameterene i IEEE 802.15.4 ikke er valgfrie i WirelessHARTs benyttelse av IEEE 802.15.4, men snarere definert som konstanter. Dette gjør også at vi kan fjerne noe overflødig funksjonalitet relatert til dynamikken i disse parameterene fra det fysiske laget i AVR2025.

Til sist, og til viss grad på grunn av det vi nettopp har omtalt, har vi valgt å omorganisere tjenesteprimitivstrukturen mellom det fysiske laget og datalinklaget i henhold til WirelessHARTs foreslåtte struktur. Dette er i praksis et forenklet sett av de tjenesteprimitivene som defineres mellom det fysiske laget og datalinklaget i IEEE 802.15.4-standarden. AVR2025-biblioteket har imidlertid ikke implementert tjenesteprimitivene slik IEEE 802.15.4 foreslår, noe vi antar skyldes den tette koblingen mellom datalinklaget og det fysiske laget i deler av funksjonaliteten. Selv om det ikke er noe i veien for dette så lenge alle tjenestene til syvende og sist blir korrekt utført, forenkler det arbeidet og oversikten å konsekvent følge WirelessHART-standardens foreslåtte tjenesteprimitivstruktur.
Kort oppsummert betyr dette tre større endringer i AVR2025-bibliotekets fysiske lag: En dekobling av enkelte MAC-funksjoner fra maskinvaren, fastsette deler av PIB-strukturen i henhold til WirelessHARTs krav, samt endre tjenesteprimitivstrukturen for bedre å passe sammen med WirelessHART-standarden og WirelessHARTs datalinklag. Under skal vi kort gå gjennom hva disse endringene innebærer.

6.3.1 Dekobling av MAC-funksjoner fra maskinvaren

Som allerede nevnt har Atmel-brikken vi benytter støtte for å utføre enkelte MAC-funksjoner, funksjoner knyttet til medietilgangskontroll, direkte i maskinvaren. Det vil si at ved å sette spesifiserte verdier i de rette maskinvareregisterne vil brikken selv ta hånd om disse MAC-funksjonene. Vi har allerede nevnt utsending av ACK og automatisk retransmisjon, og i tillegg støttes også CRC-beregning (Cyclic Redundancy Check) ved både sending og mottak, samt utføring av CSMA/CA (Carrier Sense Medium Access with Collision Avoidance).
ACK-ing i IEEE 802.15.4 er nokså trivielt, siden en ACK i denne protokollen i hovedsak bare består av å sende sekvensnummeret på den innkommende pakken i retur så snart som mulig, sammen med en ACK-rammeidentifikator og en CRC-beregning. Av denne grunn er det praktisk å gjøre dette i maskinvaren. I WirelessHART er ACK-ing derimot langt mer komplisert. Vi har allerede sett i kapittel 4.3.4 at en WirelessHART-ACK skal inneholde en responskode som forteller noe om resultatet av overføringen, samt tidsdifferansen mellom tid for mottak og forventet tid for mottak for å bidra til å opprettholde synkroniseringen i nettverket. Alt dette er verdier som krever beregning i programvaren, og som gjør at maskinvarestøtten for ACK-ing ikke er relevant for oss. Vi har derfor fjernet dette fra TAL-modulen (Tranceiver Abstraction Layer) og i stedet implementert vår egen ACK-funksjonalitet på datalinklaget.

Automatisk retransmisjon benyttes av datalinklaget i IEEE 802.15.4 der data ikke har blitt ACK-et etter en viss tid. Avsenderen vil da forsøke på nytt et antall ganger før den gir opp og gir beskjed til overliggende lag om at overføringen feilet. Også dette er implementert i maskinvaren i Atmel-brikken, men i likhet med ACK-funksjonaliteten lar det seg ikke forene med WirelessHART. Retransmisjon i WirelessHART foregår ved at mangelen på mottatt ACK ikke fører til noe annet enn at pakken ikke blir fjernet fra den utgående pakkekøen, og ansvaret for å retransmittere pakken ligger dermed på linkplanleggeren som må finne en ny, egnet tidslomme senere. Også automatisk retransmisjon er derfor fjernet fra TAL-modulen.

I motsetning til for ACK-ing og retransmisjon, benytter IEEE 802.15.4 og WirelessHART seg av den samme CRC-algoritmen, nærmere bestemt ITU-T (International Telecommunication Union - Telecommunication Standardization Sector). Dette er derfor en maskinvarestøttet algoritme vi kan dra nytte av. AVR2025-biblioteket i kombinasjon med Raven-brikken løser dette ved at alle utgående pakker automatisk får CRC beregnet og lagt på halen, slik begge standardene krever. Alle innkommende pakker får automatisk CRC beregnet på nytt og sammenlignet med CRC-verdien i pakken, før CRC-feltet fjernes. Pakken kan så, uansett resultat av CRC-beregningen, hentes ut fra maskinvaren for videre behandling. Resultatet av CRC-sammenligningen kan leses ut fra et felt i maskinvareregisteret i den videre behandlingen, og er i så måte fullt kompatibelt med WirelessHART.

Avslutningsvis tar vi med at deler av CSMA/CA-algoritmen også legges opp til å utføres i maskinvaren i AVR2025-biblioteket. Siden WirelessHART bare benytter CSMA/CA i enkelte situasjoner (tidslommer med delte linker, omtalt på side 57 i kapittel 4.3.5), men har behov for CCA (Clear Channel Assesment) som er en del av den komplette CSMA/CA-algoritmen i langt flere tilfeller (vanlige sende-tidslommer), er det nødvendig å ha tilgang til CCA fra datalinklaget. CCA benyttes her for å kontrollere om mediet er ledig, for dermed å slippe å bruke strøm på overføring av data som uansett vil feile. CCA hører hjemme på det fysiske laget, men tilgjengeliggjøres for CSMA/CA på datalinklaget gjennom tjenesteprimitiver. Dette er ikke gjort i AVR2025-biblioteket på grunn av den tette koblingen mot maskinvaren i implementeringen av CSMA/CA-algoritmen. I stedet utføres CSMA/CA med CCA automatisk ved pakkeutsending. Vi har derfor deaktivert AVR2025s CSMA/CA-håndtering, og i stedet tilgjengeliggjort CCA for datalinklaget gjennom tjenesteprimitiver. Dette innebærer at CSMA/CA-algoritmen må implementeres på nytt på datalinklaget for å kunne brukes i tidslommer med delte linker, men vi ser ikke på dette som en ulempe da CSMA/CA-

algoritmen i WirelessHART og IEEE 802.15.4 uansett ikke er identiske og dermed uansett vil kreve endringer.

6.3.2 PAN Information Base (PIB)

IEEE 802.15.4 opererer med en datastruktur kalt PIB, som vi tidligere såvidt har omtalt i kapittel 4.2.2. Denne strukturen inneholder åtte konfigurasjonsverdier for enheten protokollen kjører på:

- *phyCurrentChannel*: Kanalen overføringen skal foregå på.
- *phyChannelsSupported*: En array over hvilke kanaler som støttes av enheten.
- *phyTransmitPower*: Enhetens overføringseffekt.
- *phyCCAMode*: CCA-modiene omtalt på side 40 i kapittel 4.2.4.
- *phyCurrentPage*: Kanal-page. Ikke relevant for 2,4 GHz-båndet.
- *phyMaxFrameDuration*: Maks antall symboler i en ramme.
- *phySHRDuration*: Antall symboler i synkroniseringshodet.
- *phySymbolsPerOctet*: Antall symboler i en oktett (byte).

Som beskrevet i kapittel 3.2.2 om IEEE 802.15.4 kan det velges mellom ulike frekvensbånd som medfører ulike egenskaper knyttet til antall kanaler, overføringshastighet etc. De fleste av verdiene i PIB-strukturen avhenger av dette valget. Siden WirelessHART kun benytter 2,4 GHz-båndet i IEEE 802.15.4 PHY kan de verdiene dette gjelder derfor implementeres som konstanter: `phyChannelsSupported` vil alltid være kanal 11 til 15, `phyCurrentPage` benyttes ikke på 2,4 GHz og er derfor 0, `phyMaxFrameDuration` er 266, `phySHRDuration` er 10 og `phySymbolsPerOctet` er 2. WirelessHART praktiserer kanalhopping, og verdien `phyCurrentChannel` vil derfor endre seg fra tidslomme til tidslomme. `PhyTransmitPower` skal kunne styres gjennom «local management»-tjenesteprimitivene i WirelessHART, mens den i IEEE 802.15.4 kun er lesbar. `PhyCCAMode` skal i WirelessHART alltid være 2. I forhold til AVR2025-bibliotekets PIB-implementering medfører ikke noe av dette vesentlige endringer siden den allerede er skrevet for 2,4 GHz-båndet som har tilsvarende verdier. Vi setter CCA-modusen til 2, og nøyer oss med det i denne omgang. Derimot endres tjenesteprimitivene ganske betydelig sammenlignet med AVR2025. Vi skal se på vår implementering av disse.

6.3.3 Tjenesteprimitiver

Som beskrevet tidligere har vi valgt å implementert alle tjenesteprimitivene som rene funksjonskall. Ut fra dette har vi laget et grensesnitt mot det fysiske laget som består av alle request-funksjonene, og et grensesnitt opp mot datalinklaget fra det fysiske laget som består av alle indicate- og confirm-funksjonene. Siden standarden ikke alltid er like detaljert gir vi her en kort gjennomgang av vår tolkning av hvordan de ulike tjenesteprimitivene bør implementers, før vi ser på hvordan de brukes og virker sammen for å utføre de ulike tjenestene. Dette er også illustrert i Figur 6-2. For detaljene i implementeringen av tjenesteprimitivene henviser vi til kildekoden.

Grensesnittet det fysiske laget tilbyr datalinklaget består av de tre funksjonene `enable_request(state, channel)`, `cca_request()`, og `data_request(frame)`. Førstnevnte sjekker om radioen sover, og vekker den i så fall til liv via de eksisterende funksjonene for dette i AVR2025-bibliotekets TAL (Tranciever Abstraction Layer). Deretter settes rett kanal i PIB-en, og TAL vil samtidig sørge for at radioen innordner seg etter dette. Til sist settes radioen til enten å lytte eller sende basert på state-parameteret, også dette via ferdige funksjoner i TAL. Funksjonen avsluttes ved å kalle `enable_confirm`-funksjonen i datalinklaget med modus (lytte eller sende) og kanal som parametere. Confirm-funksjonen formidler svaret fra det fysiske laget til datalinklaget om hvorvidt radioen er klar til å sende data, og endrer TDMA-maskinens tilstand ut fra dette. CCA-request-funksjonen setter radioen i lyttemodus, ber radioen om å utføre CCA gjennom å skrive til en spesifikk CCA-registerverdi, og sjekker («poller») så radioens status inntil den sier at CCA er gjennomført. Funksjonen avslutter med å kalle `cca_confirm`-funksjonen i datalinklaget, med resultatet av CCA (om mediet er ledig eller opptatt) som parameter. Foreløpig har vi latt `cca_confirm` være en tom funksjon i MAC, men den tat en status som parameter som kan være en av tre: `TranceiverOff`, `ChannelBusy` eller `ChannelIdle`. Vi foreslår å etter hvert utvide `cca_confirm` til å endre TDMA-tilstandsmaskinen i henhold til statusen, slik at den videre programflyten kan kontrollere TDMA-tilstandsvariabelen for om det er klart for sending eller ikke. For øyeblikket blir resultatet av CCA ignorert, siden `cca_confirm` ikke er implementert.

Data-request-funksjonen utfører selve sendingen av data. Den tar en ferdig ramme (med unntak av CRC-beregningen, som gjøres i maskinvaren) som parameter, sjekker om radioen er våken, klar til å sende og ikke opptatt med en annen ramme, og hvis alt er i orden skriver dataene til radioen som videre utfører CRC-beregningene og sender rammen ut på mediet.

Uansett utfall avsluttes funksjonen med et kall på `data_confirm` i datalinklaget med en responsverdi som forteller hvordan forsendelsen gikk, samt en peker til rammen som ble sendt eller forsøkt sendt. Status er `Success` ved feilfri sending og en av `TranceiverOff`, `ReceiverOn` eller `TransmitterBusy` ved feil i sending.

Indicate-tjenesteprimitivene i det fysiske laget informerer datalinklaget om hendelser som skjer nedenfra og opp. Siden vi implementerer primitivene som funksjonskall er disse derfor lagt «nederst» i datalinklaget og eksponert til det fysiske laget som et grensesnitt. Tre indicate-funksjoner er tilgjengelig for det fysiske laget; `enable_indicate()`, `data_indicate(rsl, data)`, og `error_indicate(status, data)`. Hva som skjer etter at funksjonen blir kalt er opp til datalinklaget, og blir derfor ikke gjennomgått før senere. I stedet ser vi på hva som leder ut i disse kallene: Førstnevnte skal gi datalinklaget beskjed om at en pakke nettopp har begynt å komme inn på det fysiske laget, og kalles derfor umiddelbart etter at radioen merker at data er på vei inn. Så vidt vi har kunnet finne ut sender ikke radioen et avbrudd (interrupt) når dette skjer, og det er derfor ikke mulig for oss å utføre dette funksjonskallet til riktig tidspunkt. Vi har likevel implementert funksjonen, men i praksis vil dette kallet alltid bli utført umiddelbart før `data_indicate()`-kallet, som på sin side informerer datlinklaget om at hele rammen er mottatt. Dette skyldes at vi på mottakersiden først mottar et avbrudd når hele rammen er mottatt, og dermed ikke vet at data er på vei inn før dette. I praksis har ikke dette den store betydningen, siden `enable_indicate()` kun skal føre til at datalinklaget registrerer tidspunktet for start av mottak. Siden vi kjenner tiden det tar å overføre en oktett (byte), og dette kallet blir utført rett etter at mottak er ferdig, finner vi tidspunktet for starten ved mottak med følgende formel:

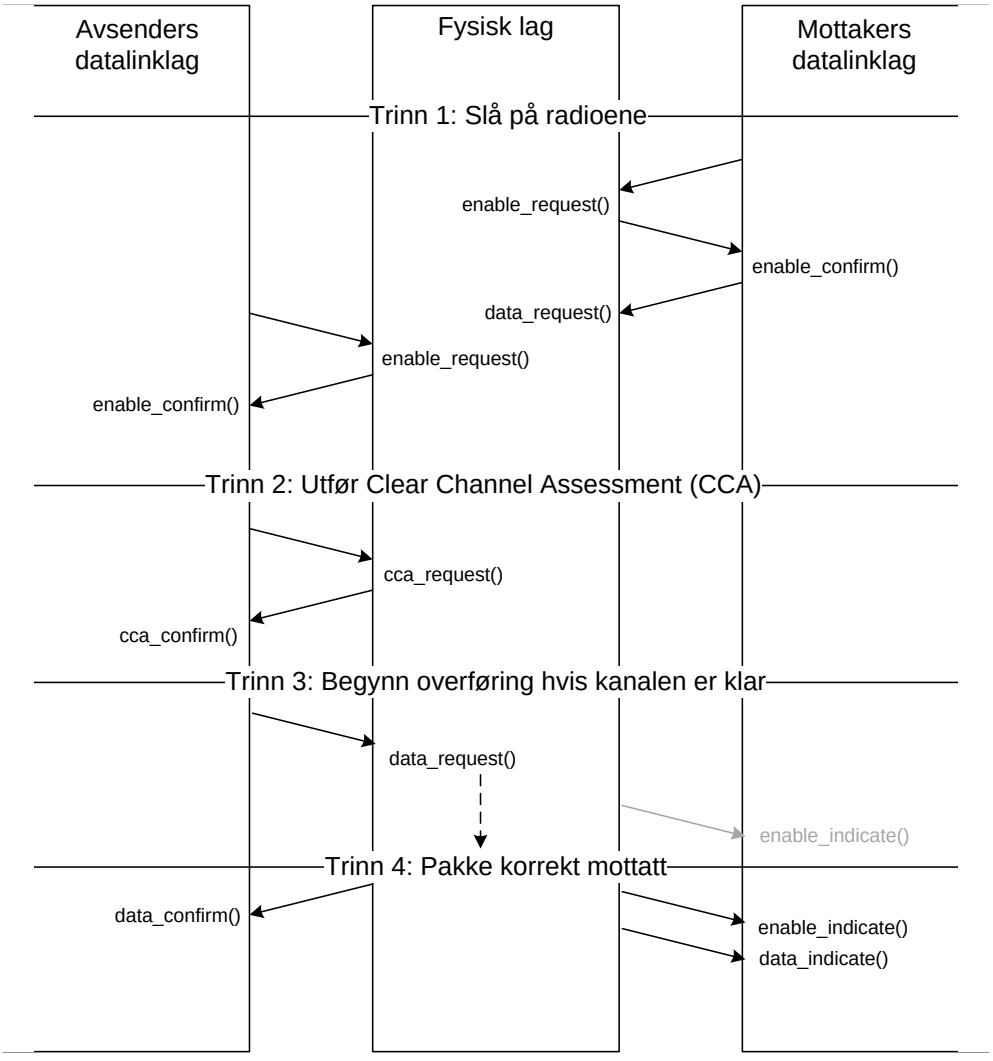
$$start\_of\_recv\_time = now\_time - (octet\_transmit\_time * frame\_length)$$

Funksjonen `error_indicate()` utstedes ved feil i mottak eller behandling av mottatt data på det fysiske laget, og dette kallet utføres i vår implementering i de tilfeller hvor den mottatte rammen overgår grensen for maksimal rammestørrelse.

Ut over dette legger spesifikasjonen opp til en administrasjonstjeneste, der ulike verdier som overføringseffekt og radiostatus kan styres. Vi har ikke implementert dette etter standarden, men benytter oss foreløpig av tilsvarende, eksisterende funksjonalitet i TAL-modulen. Unntaket er muligheten for å styre bruken av ERROR.indicate-tjenesteprimitivet, som ikke er

til stede i vår implementering. Vi anser dette som en detalj som ikke er viktig på det nåværende stadiet av utviklingen, men som må implementeres i en ferdig protokoll.

Figur 6-2 viser hvordan vi kombinerer de ulike tjenesteprimitivene i en normal dataoverføring. Disse hendelsene initieres fra datalinklaget gjennom TRANSMIT-tjenesten, og vi skal gå grundigere gjennom hvordan gangen i dette er implementert senere, men det kan likevel være greit å få et innblikk i dette allerede nå.



Figur 6-2 Gangen på det fysiske laget i en normal dataoverføring

Det fysiske laget er tilsynelatende felles for de to nodene fordi selve overføringsmediet er abstrahert vekk. I realiteten ligger det fysiske laget på hver sin node. Alle hendelsene vist i figuren skjer i løpet av én tidslomme, og starter ved at mottaker setter sin radio i lyttemodus og avsender setter sin radio i sendemodus. I følge spesifikasjonen skal mottaker også sende en DATA.request-primitiv etter at radioen er bekreftet klar, men det er uklart for oss hva

hensikten med dette er. I avsendermodus gjøres dette for å sende en datapakke ned til det fysiske laget og videre ut på overføringsmediet. Siden parameteret er en dataramme antar vi at hensikten i mottakssammenheng er å sende et tomt, klargjort buffer til det fysiske laget som den innkommende pakken kan lagres i. I vår implementering sørger det fysiske laget for å initiere et slikt buffer selv, siden alle lagene uansett deler bufferhåndteringsmodulen fra AVR2025-biblioteket, og kallet på `data_request`-funksjonen har derfor ingen effekt hvis radioen står i sendemodus. Vi har likevel lagt inn dette kallet, siden det er slik spesifikasjonen beskriver gangen i en overføring og dermed lagt til rette for en senere tilpasning til standardens design.

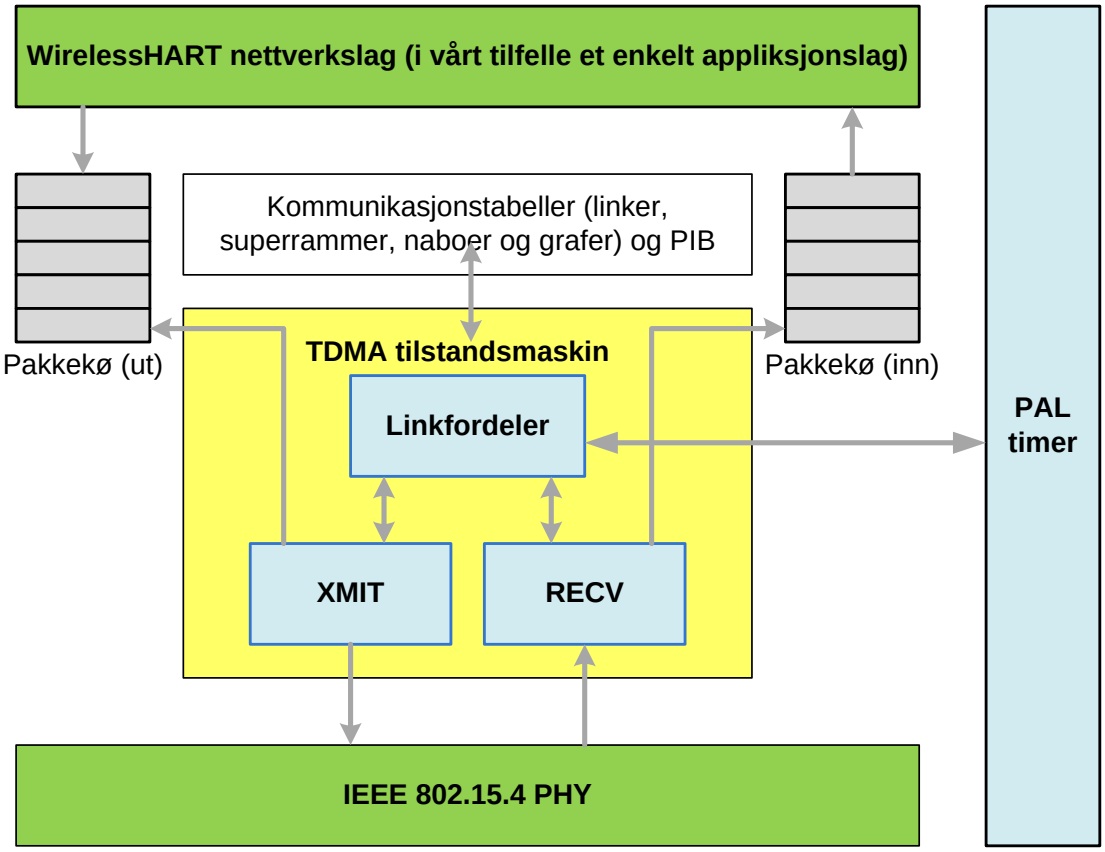
Trinn to iverksettes av avsender for å verifisere at mediet er ledig. Ved positivt svar kan dataoverføringen settes i gang, slik trinn tre viser. Vi ser også at mottakers fysiske lag i følge spesifikasjonen skal utstede en ENABLE.indicate-primitiv for å informere om at data har begynt å komme inn på radioen, men at dette først skjer i trinn fire i vår implementering og er derfor grånet ut.

6.4 Datalinklaget

Som vi allerede har vært inne på ved flere anledninger, er WirelessHARTs datalinklag svært ulikt datalinklaget i IEEE 802.15.4. Vi forsøkte innledningsvis å beholde IEEE 802.15.4-datalinklaget som var implementert i AVR2025-biblioteket for så å gradvis tilpasse dette til WirelessHART, men så etter hvert at dette ikke var hensiktsmessig. I stedet har vi derfor utviklet begynnelsen på et datalinklag fra bunnen av. Arkitekturen, slik vi har implementert den, kan i grove trekk illustreres som i Figur 6-3. Dette er en figur som er sterkt inspirert av forslaget til datalinklagsarkitektur presentert i Song et al [49]. Vi kan gruppere komponentene inn i en todeling som korresponderer til henholdsvis LLC-laget (kommunikasjonstabeller og PIB, samt pakkekøene) og MAC-laget (TDMA-maskinen). Dette er den logiske oppdelingen av datalinklaget som WirelessHART-spesifikasjonen benytter. Merk at PIB (PAN Information Base) ikke er den samme strukturen som PIB-en vi omtalte i forbindelse med det fysiske laget, men en tilsvarende struktur med datalinklagsrelevant konfigurasjonsinformasjon. Vi vil komme tilbake til denne strukturen senere.

WirelessHART-nettverkslaget er tegnet inn på figuren, men er ikke implementert. I stedet har vi laget et svært enkelt applikasjonslag som vi har lagt på toppen av datalinklaget. Dette applikasjonslaget har i all hovedsak bare én oppgave: Lage en dummy-datapakke og sende

den ned til datalinklaget via datalinklagets tjenesteprimitiver én gang hvert N-te sekund. Dette gjøres for den senere testingen og visualiseringen.



Figur 6-3 Datalinklagets oppbygning

Vår implementering av datalinklaget er på ingen måte komplett, og viktige elementer som en fullstendig tidssynkronisering, håndtering av graf-rutede nettverkslagspakker, WirelessHARTs sikkerhetsmekanismer, samt håndtering av konfigurasjonsmeldinger fra nettverksadministrator er ikke implementert. Vi mener likevel at strukturen presentert i Figur 6-3 vil være gyldig også i et komplett datalinklag, og har i vår implementering forsøkt å ta hensyn til senere, nødvendige utvidelser. I de neste underkapittelene vil vi gå gjennom de elementene av datalinklaget som helt eller delvis *er* implementert, og forsøke å begrunne de valg vi har gjort. Vi begynner fra toppen med tjenestene som tilbys nettverkslaget, før vi går dypere inn i henholdsvis LLC-laget og MAC-laget.

6.4.1 Tjenesteprimitiver

Som tidligere beskrevet tilbyr datalinklaget flere tjenester til nettverkslaget, bestående av en rekke tjenesteprimitiver. Vi har fokusert på å få til en fungerende dataoverføring, og derfor kun implementert overføringstjenesteprimitivene TRANSMIT.request, TRANSMIT.confirm og TRANSMIT.indicate i denne omgang. I likhet med tjenesteprimitivene på det fysiske laget er disse implementert som rene funksjonskall, i motsetning til AVR2025-bibliotekets datastruktur- og hendelseskø-design. Spesifikasjonen beskriver fire varianter av TRANSMIT.request, avhengig av om dataene skal graf-rutes, kringkastes, sendes til en spesifikk kort adresse, eller sendes til en spesifikk lang adresse. Vi har slått dette sammen til en enkel `transmit_request()`-funksjon, som alltid mottar mottakerinformasjonen som en 64-biters integer sammen med et flagg som forteller hva slags adresse det er snakk om. Dermed kan mottakerinformasjonen relativt enkelt tolkes til enten en graf-ID, kringkastingsadressen, en kort adresse eller en lang adresse avhengig av flaggets verdi, og datalinklaget handle deretter. Med alle nødvendige parametere ender vi opp med et funksjonskall som dette:

`transmit_request(handle, payload, length, priority, timeout, adress_mode_flag, destination_info, bcast_sframe).`

Det siste parameteret leses bare i tilfeller hvor `address_mode_flag` er kringkasting, og angir hvilken superframe som skal benyttes for kringkastingen. Funksjonen har mange parametere, og noen av dem, for eksempel de tre relatert til mottakeraddresseringen, kan nok med fordel samles i en datastruktur for å lette oversikten.

Oppgaven til request-funksjonen er først og fremst å legge en nettverkspakke på datalinklagets utgående kø. Selve sendingen vil bli tatt hånd om av linkfordeleren i etterkant. Siden elementene på denne køen også skal inneholde en del metainformasjon, har vi implementert en datastruktur som tar hånd om dette. Standarden trekker frem pakke-ID, selve pakken, mottakerinformasjon og tidspunktet for når pakken ble køet som parametere som må tas vare på i både inn- og utkøen. Vi har, av implementeringstekniske årsaker, også tatt med en del andre verdier i denne strukturen. For eksempel har vi på dette stadiet all nødvendig informasjon for å generere DLPDU-spesifikatoren og adressespesifikatoren til den endelige datalinklagspakken, og lager derfor disse allerede her og legger dem på den køede strukturen. Dette for å raskere kunne generere den ferdige datalinklagspakken ved sendetidspunktet, der vi har en relativt knapp timing å forholde oss til.

Vår implementering av TRANSMIT.request er, kort oppsummert, omtrent slik i pseudo-kode:

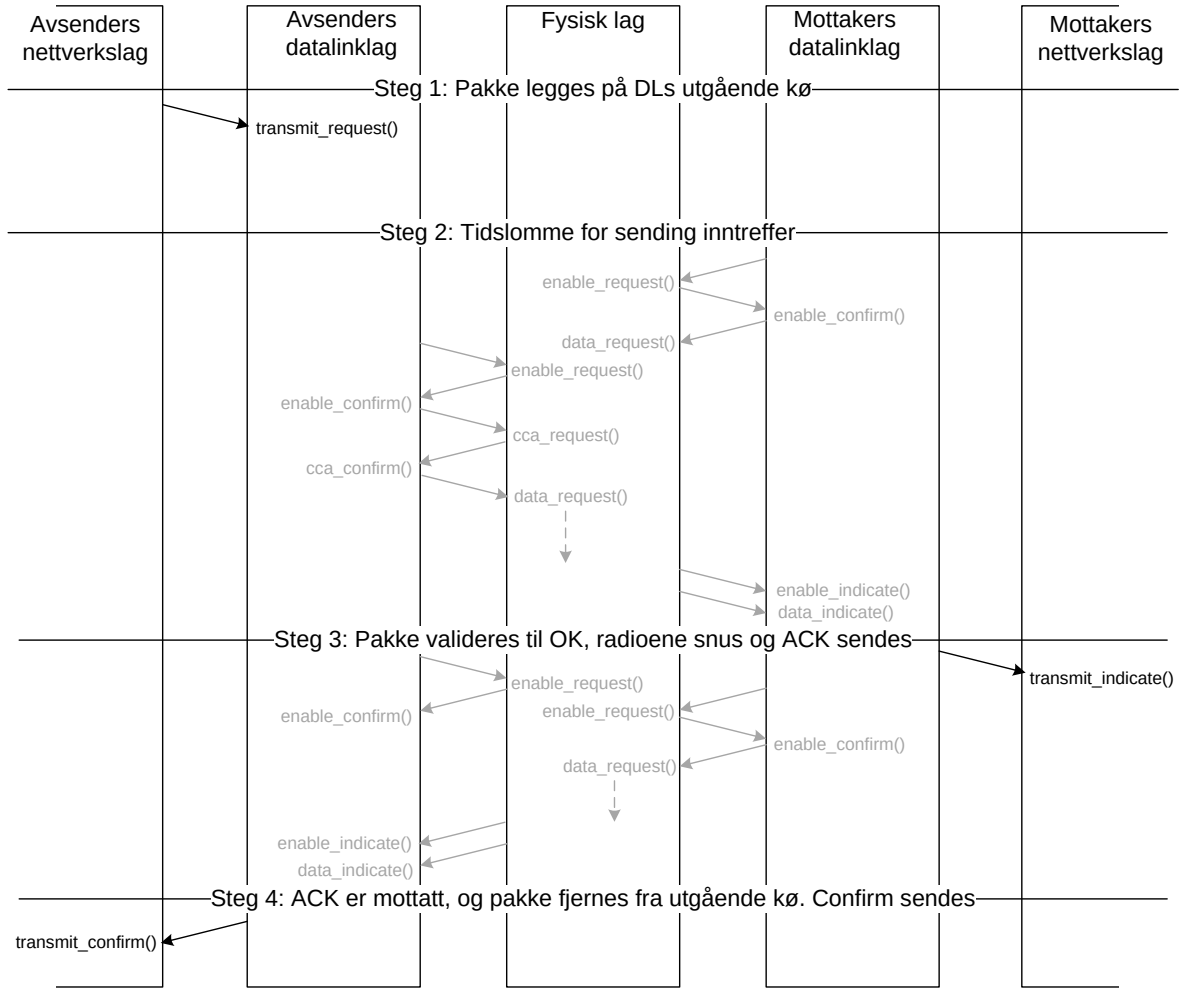
```
transmit_request(…)
{
      * Generer DLPDU-spesifikator
      * Generer addressespesifikator
      * Alloker datastrukturen som skal køes, og fyll den med
        nettverkspakken og metainformasjonen, herunder de to genererte
        spesifikatorene
      * Skriv tidspunkt til timestamp-variabelen i datastrukturen

      if (queue_append(&transmit_queue, packet_record) != SUCCESS) {
            * utfør en TRANSMIT.confirm med feilmelding
      }

      link_scheduler();
}
```

Vi ser at funksjonen avsluttes med et kall på linkfordeleren. Dette fordi en overføringsforespørsel er en hendelse av en slik art at en ny linkfordeling er påkrevet; den nye pakken kan ha rokket ved den rekkefølgen linkfordeleren planla ved forrige fordeling. Vi vil diskutere dette i større detalj i avsnittet om implementeringen av vår linkfordeler senere, og i den forbindelse vil det også komme frem at det er viktig å raskt kunne plukke ut hvilke pakker som skal til hvilke mottakere i linkfordeleren. Vi har derfor sett på muligheten for å lage en utkø for hver enkelt mottakerlink, i stedet for én stor, felles utgående kø. Dette vil gi vesentlige besparelser til linkfordeleren som vil slippe unna en rekke iterasjoner over hovedutkøen for å se om hver enkelt link har ventende pakker. I stedet kan den gå rett på den aktuelle linkens egen kø og se om den er tom eller ikke. Imidlertid innebærer graf-ruting, som er WirelessHARTs foretrukne ruting, at valget av hvilken link en pakke skal sendes ut på ikke avgjøres før ved selve linkfordelingen. I køingsprosessen vet man i tilfellet med graf-ruting bare listen over alternative naboer som kan velges for den gitte graf-ID-en. Hvordan skal pakken køes i slike tilfeller? Vårt forslag er å køe pakken på køene til *alle* linkene som ligger i den aktuelle grafens liste over potensielle nabolinker. Tilsvarende med kringkastingspakker: Disse pakkene køes på alle linkene tilhørende superrammen nettverkslaget har bedt om å kringkaste på. I begge tilfeller må pakken fjernes fra samtlige køer ved første vellykkede utsending (første utsending ved kringkasting, første mottatte ACK ved graf-ruting). En slik strategi fører til noe ekstra minnebruk, da hver pakke må legges i en ny listenodedatastruktur

(`struct`) for hver kø den skal legges på. Med en enkel hovedutkø ville det i stedet holde med én enkelt listenodedatastruktur per pakke. I tillegg krever denne tilnærmingen ekstra tid og ressurser ved hver TRANSMIT.request fordi det blir mer jobb med fordelingen på de ulike køene, samt ekstra tid og ressurser ved vellykket utsending fordi pakken må fjernes fra alle køene den ligger på. Til gjengjeld forenkles linkfordelingsalgoritmen vesentlig, og vår oppfatning er at en slik løsning vil gi besparelser totalt, siden linkfordeling er en hyppig forekommende affære. Her gjenstår det imidlertid noe implementeringsarbeid, og ikke minst en grundigere analyse, så vi utelukker på ingen måte at denne løsningen ikke er den beste.

Ved ferdig utført pakkeutsending skal TRANSMIT.confirm-primitivet utstedes til nettverkslaget, og den videre behandlingen er deretter utenfor datalinklagets domene. Tilsvarende med TRANSMIT.indicate på mottakersiden ved utført pakkemottak. Vi har derfor implementert disse som de to tomme funksjonene `transmit_confirm(...)` og `transmit_indicate(...)` i det som for oss er applikasjonslaget. De har altså ingen funksjon i vår implementering, men legger til rette for en implementering av et WirelessHART-nettverkslag på et senere tidspunkt. Indicate-funksjonen skal kalles for hvert *suksessfulle* mottak av en pakke, mens confirm-funksjonen også kan bli kalt når utsending av data feiler. Feil eller suksess angis i et status-parameter i funksjonen.

Sammen med den kunnskapen vi har om tjenesteprimitivene i det fysiske laget fra forrige underkapittel, kan vi nå danne oss et bilde av meldingsutvekslingen internt i nodene i en overføring som begynner i nettverkslaget i én (nabo)node og ender i nettverkslaget til en annen (nabo)node. Figur 6-4 illustrerer dette.



Figur 6-4 Gangen i datalinklaget og det fysiske laget i en normal dataoverføring

De grånede funksjonskallene tilhører det fysiske lagets tjenesteprimitiver og er vist tidligere i Figur 6-2, men den gang uten ACK. Legg for øvrig merke til at CCA (Clear Channel Assessment) ikke utføres ved ACK-ing; dette er helt etter spesifikasjonen. Det antas at mediet fremdeles er ledig siden vi allerede utførte CCA tidligere i tidslommen.

Det bør bemerkes at hendelsesforløpet ovenfor illustrerer feilfri overføring av en normal datapakke til en nabo. Andre forløp kan være følgende:

- Ved full utkø på avsendersiden vil `transmit_confirm()` bli kalt umiddelbart med feilmelding.
- Ved fullt mottaksbuffer på mottakersiden, vil ACK-en gi beskjed om dette, og datalinklaget vil utføre retransmisjoner. Retransmisjoner skjer også hvis ACK ikke

mottas i det hele tatt. Hvis terskelen for antall retransmisjoner overstiges eller pakken timer ut (omtalt i avsnitt om lagring av pakker, side 58, kapittel 4.3.5) vil `transmit_confirm()` bli kalt med beskjed om dette.

- Hvis pakken som skal sendes ut er en kringkastingspakke skal ikke ACK forventes, og `transmit_confirm()` kalles derfor umiddelbart.

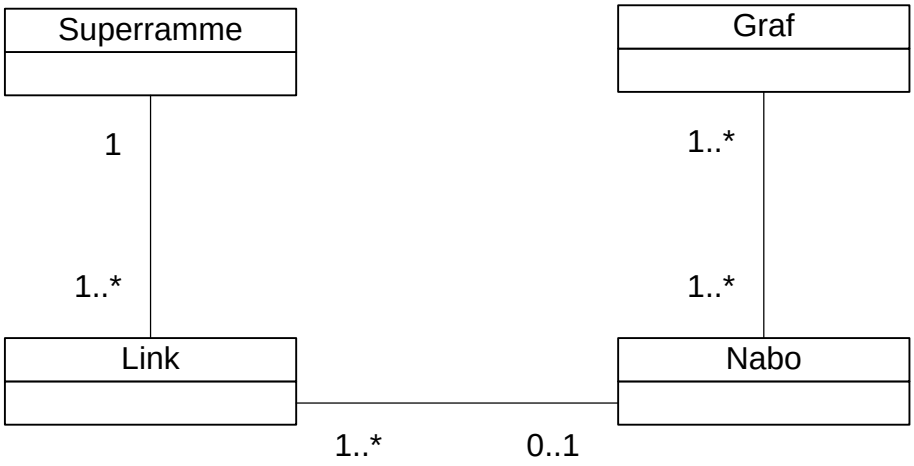
6.4.2 LLC-sublaget (Logical Link Control)

I kapittel 4.3 gikk vi gjennom spesifikasjonen av datalinklaget i WirelessHART, og vi så at en mengde informasjon må lagres i hver node for å opprettholde driften og utføre lagets oppgaver på en korrekt måte. Dette er informasjon om tilknyttede linker, naboer, grafer og superrammer. I tillegg til å holde orden på denne informasjonen er det LLC-sublagets oppgave å konstruere de ulike DLPDU-typene (Data-Link Protocol Data Unit) på forespørsel. Hver DLPDU-type har sin egen «oppskrift» på hva den skal inneholde, og LLC-sublaget har ansvaret for å generere pakkene i henhold til disse. Ut over dette må datalinklaget ha oversikt over en del informasjon om seg selv, slik som for eksempel sin egen korte og lange adresse. Spesifikasjonen sier lite om hvordan dette bør gjøres, og vi har derfor konstruert en datalinklags-PIB (PAN Information Base) som håndterer dette, etter inspirasjon fra Song et al [49]. Vi skal videre i dette underkapittelet gå gjennom de valg, utfordringer og løsninger vi har vært innom i implementeringen av alle disse elementene i LLC-sublaget.

Datastrukturer

Vi har allerede gått gjennom hva standarden krever av informasjon som skal lagres i de ulike datastrukturene for linker, naboer, grafer og superrammer (kapittel 4.3.5), og siden vi har overført denne informasjonen mer eller mindre direkte til C-structer i vår implementering avstår vi fra å gå gjennom disse i detalj. Den eneste vesentlige endringen fra standarden er at vi har erstattet det som tilsynelatende er bruk av arrayer med lister. Her kan det uansett virke som om bruken av array-notasjonen i standarden bare er ment å indikere «elementsamling», og ikke nødvendigvis at det skal implementeres som en array. Vi innser at en liste kan gi noe høyere minnebruk siden alle elementene må «pakkes inn» i node-strukturer, men til gjengjeld forenkler det sortert uttak og innsetting av elementer. Dette har betydning i forbindelse med linkfordeling der sortering (som vi vil se i diskusjonen rundt vår implementering av linkfordeleren senere) er sentralt. Blant annet må vi sortere tidslommer i rekkefølgen de inntreffer, sortere pakker etter tidspunkt for køing, samt sortere pakker etter prioriteter. Lister

tillater også dynamisk størrelse på elementsamlingene på en enkel måte, noe som ofte er komplisert med arrayer. Når dette er sagt så er muligens dette et punkt som fortjener en grundigere analyse ved en komplett implementering av datalinklaget. I vår sammenheng har slike spørsmål liten betydning, men i et nettverk hvor den samlede mengden listeelementer kan bli betraktelig mye større kan tidsforbruket i verste fall bli for høyt til at protokollen kan utføre jobben sin innenfor den strikte timingen som WirelessHART krever, og i alle tilfeller er det av interesse å ha effektive algoritmer som kan holde tids- og energiforbruket nede.
Selv om vi som nevnt ikke går i detalj på datastrukturene, kan det være av nytte for den videre forståelsen å se hvordan de henger sammen. Figur 6-5 viser dette i en forenklet UML-notasjon.



Figur 6-5 UML-notasjon av forholdet mellom strukturene superramme, link, nabo og graf

Vi ser at hver superramme har en liste over alle linker som tilhører superrammen, og at hver link bare skal kunne tilhøre én enkelt superramme. Hver link har en referanse til én enkelt nabo, eller kan også være en delt link eller kringkastinslink og derfor ikke ha referanse til noen spesiell nabo. Hver graf har på datalinklaget en liste over naboer som kan være potensielle neste hopp. Figuren i standarden som denne figuren er basert på opererer med en ener-multiplisitet fra nabo til link og fra nabo til graf, men vi mener at dette ikke stemmer overens med standarden for øvrig og at det i stedet må være «1..*»-multiplisiteter her. Dette fordi en nabo skal kunne tilordnes flere linker i samme superramme hvis det er behov for hyppigere kommunikasjon enn bare én tidslomme per superramme, og fordi en nabo fint kan være et mulig neste-hopp i mange ulike grafer. Siden relasjonene uansett er enveis kan

imidlertid disse to multiplisitetene ignoreres i implementeringen, og har derfor ikke den store praktiske betydningen.

PIB (PAN Information Base)

I tillegg til datastrukturene vist i Figur 6-5 har vi valgt å implementere en PIB-struktur på datalinklaget for å samle all den relevante og påkrevde informasjon om noden. Dette er, som tidligere nevnt, ikke en struktur som benyttes i standarden, men i stedet foreslått av Song et al [49]. En alternativ måte å gjøre det på ville være å oppbevare informasjonen i globale konstanter og variabler. At strukturen kalles for PIB henger sammen med bruken av dette begrepet i IEEE 802.15.4-standarden, og legger ingen spesielle føringer i implementeringen utenom en forventning om at den skal lagre nettverksrelatert informasjon. Vår PIB på datalinklaget holder orden på enhetens korte og lange adresse, nettverks-ID og et flagg som indikerer hvorvidt enheten er autentisert i nettverket eller ikke. Vi ser ikke bort i fra at det kan bli aktuelt å legge mer data inn i denne ved videre implementering.

DLPDU-konstruksjon

Som tidligere nevnt er et av LLC-lagets hovedoppgaver i WirelessHART å generere de ulike DLPDU-typene på forespørsel. Dette kan være vanlige datapakker, ACK-pakker, keep-alive-pakker, advertise-pakker og frakoblingspakker, alle tidligere beskrevet i kapittel 4.3.4. Siden vi kun har fokusert på å få vanlig pakkeoverføring til å fungere har vi nøyd oss med bare å implementere genereringen av datapakker og ACK-pakker. Datapakkegeneratoren tar en datastruktur fra pakkekøen som parameter, tolker informasjonen i denne og kopierer eller konstruerer de rette verdiene til et tomt buffer. Til slutt sitter vi igjen med en fullverdig DLPDU. I ACK-generatoren tar vi en datastruktur fra den innkommende pakkekøen (altså en «ferdigtolket» innkommende pakke), en responskode og en tidsverdi. Ut fra den innkommende pakken kan vi lese ut adresseringsfeltene og bytte om på disse og skrive til det tomme pakkebufferet sammen med de andre pakkehodeverdiene. Responskoden inneholder informasjon om hvorvidt pakkemottaket var vellykket, og skrives til ACK-pakkens nyttelast. Det gjør også tidsverdien, som beskriver tidsavstanden mellom forventet mottak av pakke og reelt mottak av pakke.

6.4.3 MAC-laget (Medium Access Control)

MAC-laget danner den nedre delen av datalinklaget, altså den delen som håndterer enhetens tilgang til mediet. I WirelessHART gjøres dette ved hjelp av en TDMA-protokoll (Time Division Medium Access), og en presis håndtering av hendelser og tidspunkter er derfor av avgjørende betydning. Sentralt i dette står en såkalt TDMA-tilstandsmaskin, som holder orden på hvilken tilstand MAC-laget er i og fordeler linker til tidslommer gjennom den tilknyttede linkfordeleren. Vi har tidligere sett på de teoretiske sidene ved dette, og Figur 4-8 på side 66 kan fungere som en grei oppsummering av TDMA-tilstandsmaskinen fra et overordnet perspektiv. I dette underkapitlet skal vi se på og gå gjennom de implementeringstekniske spørsmålene rundt linkfordeleren og den videre behandlingen av linkene i etterkant.

Linkfordeleren

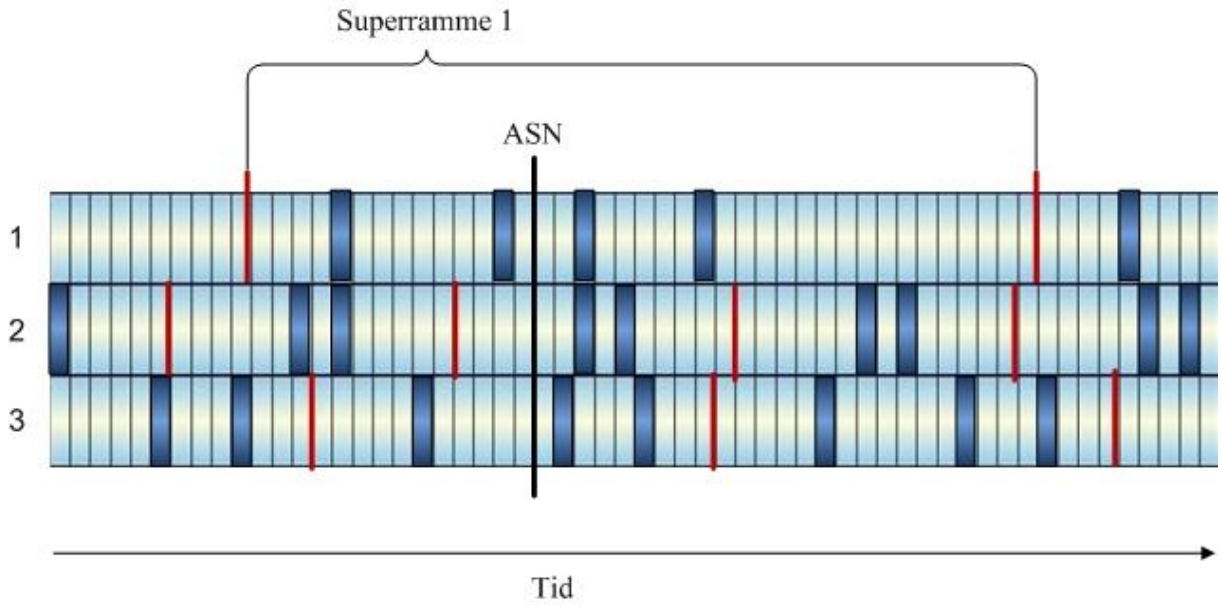
Linkfordeleren er et sentralt element i WirelessHARTs datalinklag (som beskrevet på side 60 i kapittel 4.3.5). Det er denne som avgjør hvilken tidslomme og hvilken link som til enhver tid er den neste som skal behandles, og deretter sørger for at enheten klargjøres for denne behandlingen. Det vil si at noden settes opp til sending eller mottak av en pakke på tidspunktet den utvalgte tidslommen starter på.

Hvilken link som skal behandles er ikke alltid så lett å avgjøre, siden WirelessHART byr på mange kompliserende faktorer. For eksempel kan superrammer slås av og på, legges til og fjernes, og linker kan endres. Dette er hendelser som vil kunne påvirke hvilken link som skal behandles først, og derfor må linkfordeleren kjøres på nytt ved slike tilfeller (som omtalt i kapittel 4.3.5).

Standarden gir ingen detaljert beskrivelse av hvordan linkfordeleren bør implementeres, og veldig mye er derfor opp til utvikleren. Som i WirelessHART generelt, er imidlertid energieffektivitet og tidsbruk viktig. Linkfordelingen vil normalt skje mot slutten av en tidslomme i perioden etter mottak eller utsending av en ACK. Siden neste tidslomme som skal behandles kan være den påfølgende tidslommen, må linkplanleggeren være ferdig i inneværende tidslomme (vi kommer tilbake til hva dette innebærer av tid). Vi har foreslått og implementert en algoritme som vi mener tar hensyn til dette, og som vil bli nærmere gjennomgått videre i dette underkapitlet.

Det endelige målet til linkplanleggeren er å finne og planlegge den første av tidslommene som er i ferd med å inntreffe som har en aktivitet knyttet til seg. Altså at linken tilknyttet tidslommen enten har en pakke klar til avsending, eller skal lytte etter mulig innkommende data. Vi har valgt å dele dette opp i to steg. Første punkt er å konstruere en sortert liste av

linker som vi mener kan være aktuelle for behandling. Det vil si at listen starter med de første linkene fra alle superrammene som inntreffer senere enn nåværende tidslomme. Som vi vet fra tidligere har noden oversikt over alle superrammer som benyttes i nettverket, og hver av disse superrammene har en mengde av linker knyttet til seg som hver inntreffer på en unik og spesifikk tidslomme innenfor superrammen. Dette høres kanskje mer komplisert ut enn det i realiteten er, så la oss se på en illustrasjon (Figur 6-6) som forhåpentligvis forenkler forståelsen noe.



Figur 6-6 Linkfordeling i nettverk med tre strømmer av superrammer

Illustrasjonen viser et eksempel på en situasjon der vi har tre superrammer i nettverket, i dette tilfellet kalt superramme 1, 2 og 3. Hver superramme gjentas fortløpende langs tidsaksen. Den første, superramme 1, er 39 tidslommer lang og har fire linker knyttet til seg, den andre er 14 tidslommer lang og har to linker, og den tredje er 20 tidslommer lang og har tre linker. Sett at vi nå befinner oss i linkfordelingsfasen; streken for ASN (absolute slot number) vil da markere starten på tidslommen hvor linkfordelingen ble startet fra. Med andre ord viser streken hvilken tidslomme vi befinner oss i akkurat nå. ASN kan til enhver tid finnes ved å dele antall millisekunder siden nettverksetableringen på tidslommenes lengde (som alltid er 10 millisekunder i WirelessHART). Første punkt er altså å finne ASN, og deretter ut fra ASN finne de ulike tidslommenummerene relativt til superrammens starttidslomme. Følgende formler benyttes:

ASN = nettverksalder i millisekunder / tidslommelengde

Tidslommenummer i superramme = ASN % antall tidslommer i superamme

Her kan det diskuteres hvorvidt det er fornuftig å alltid beregne ASN og tidslommenummer på denne måten. Siden vi var usikre på hvor kostbart det var å utføre divisjon på store tall på en liten brikke gjorde vi en beregning på brikken. Vi utførte de to regnestykkene over og kom til at tiden brikken benyttet var så lite som seks mikrosekunder, og vi anser dette som tilstrekkelig lavt. Om beregningene hadde tatt betraktelig lenger tid kunne vi sett på om man kunne løst problemet ved å vedlikeholde noen globale parametere.

I tilfellet som illustrasjonen beskriver ville tidslommenummerene i de tre superrammene bli henholdsvis 14, 4 og 11, og dermed vil linkene på tidslomme 16, 6 og 12 fremstå som de mest aktuelle kandidatene for neste håndtering. Særlig den nederste superrammens tidslomme 12, som er den som inntreffer på den laveste ASN og dermed nærmest i tid. Hvorvidt det faktisk blir denne linken og tidslommen som «plukkes ut» av linkfordeleren er imidlertid ikke avgjort. Vi skal komme tilbake til denne videre prosessen om et øyeblikk, men nøyer oss i første omgang med å konstatere at det er en viss mulighet for at hverken denne eller de andre linkene som ligger først i køen hos de ulike superrammene skal behandles. En avveining blir derfor hvor mange kandidatlinker man skal ta vare på fra de ulike superrammer for videre evaluering. Her ser vi i utgangspunktet tre muligheter å gjøre det på:

- Ta vare på de *N neste linkene* som potensielt skal behandles fra hver aktive superramme i en samlet liste, sortert etter ASN de eventuelt vil opptre på. Kjør linkplanleggeren på nytt hvis det viser seg at ingen av dem skulle behandles likevel.
- Ta vare på *linkene som opptrer innenfor de N neste tidslommene* som potensielt skal behandles fra hver aktive superramme i en samlet liste, sortert etter hvilket ASN de eventuelt vil opptre på. Kjør linkplanleggeren på nytt hvis det viser seg at ingen av dem skulle behandles likevel.
- Ta vare på *samtlige linker* fra hver aktive superramme i en samlet liste, sortert etter ASN de eventuelt vil opptre på. Her vil det alltid være linker som skal behandles.

Den første løsningen er den enkleste å implementere, men er opplagt ikke god. Sett at vi henter bare den første linken som opptrer hos hver av superrammene, og at bare den av dem som opptrer sist i tid viser seg faktisk å skulle behandles. Da vil ingenting skje før denne

linken blir behandlet på sin tildelte tidslomme, og vi forblir uvitende om de andre superrammene i mellomtiden muligens skulle ha fått behandlet sin «nummer to-link» (eller nummer tre, fire osv). Tilsvarende fenomen kan inntreffe uansett størrelse på N.

Den andre løsningen vil alltid gi korrekt resultat, men kan føre til mange iterasjoner hvis vi setter N for lavt, og mye sortering hvis vi setter N for høyt. Hvis N er for lav er det en god mulighet for at det ikke vil intreffe tidslommer med linker som skal behandles i det hele tatt, og hele operasjonen må gjøres på nytt med de N neste tidslommene. Hvis N er for høyt kan vi sitte igjen med veldig mange potensielle linker som må sorteres. Begge deler kan potensielt gi høyt energi- og tidsforbruk.

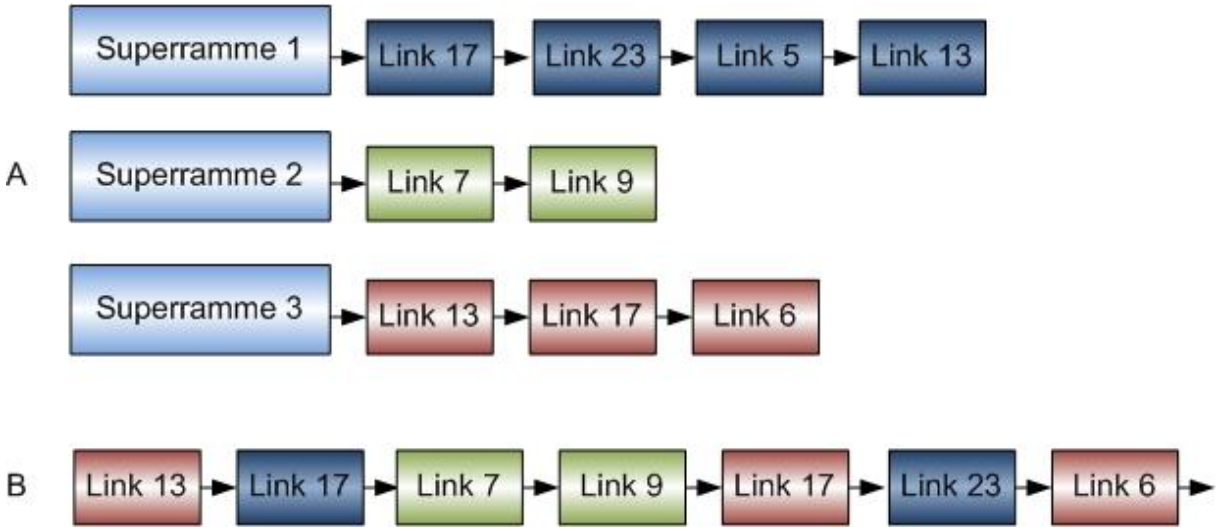
Den siste løsningen vil også alltid gi korrekt resultat, men i nettverk med mange superrammer og med mange linker i hver superramme blir dette fort en tidkrevende og lite energieffektiv øvelse fordi det krever mye sortering.

Siden alle disse tilnærmingene viser seg problematiske, legger vi til en variant til. I stedet for å plukke ut aktuelle linker for så å vurdere om de trenger behandling, kan vi heller plukke ut bare de linkene som vi vet trenger behandling. Dermed får vi denne varianten:

- Ta vare på den *første linken som* skal *behandles* fra hver aktive superramme i en samlet liste, sortert etter ASN de vil opptre på.

Vi har ikke gjennomført en grundig analyse av hvorvidt denne løsningen faktisk er den beste. For eksempel er det fullt mulig å forbedre de to forrige variantene ved å forholde seg til én sortert liste av aktuelle linker og tidslommer per superramme (representert ved variant A i Figur 6-7, basert på situasjonen vist i Figur 6-6) framfor en samlet sortert liste for samtlige superrammer (representert ved variant B i Figur 6-7). Disse listene vil være raske å beregne da superrammene allerede har en liste over sine linker sortert på det relative tidslommenummeret de inntreffer på. Til gjengjeld må det en del håndtering til for å finne ut hvilken av de ulike superrammene som har den linken som først skal behandles; det holder i mange tilfeller ikke å bare sjekke det første elementet i hver liste og se hvilken av disse som inntreffer først (her møter vi samme problematikk som beskrevet i forbindelse med punkt én i listen over mulige løsninger ovenfor). Som sagt har vi ikke rukket å gjennomføre en grundig analyse av disse spørsmålene, og som vi har vært inne på tidligere har vi generelt sett fokusert mer på å få bygget inn funksjonalitet fremfor effektivitet og hurtighet. Siden linkfordeleren

kjøres ofte og derfor er viktig for datalinklagets ytelse vil vi imidlertid anbefale at en del tid vies til slike spørsmål i en fullstendig implementering av WirelessHART.



Figur 6-7 Varianter av listehåndtering rundt linkfordeling

Vi har som nevnt valgt en løsning der vi tar vare på den første linken som skal behandles fra hver aktive superramme for så å plukke ut den med den laveste ASN for videre behandling. Hva er det så som avgjør om en link er «behandlingsklar»? Hvis linken er en mottakslink er det enkelt: Alle mottakslinker skal i utgangspunktet alltid behandles. Hvis linken er sendelink er det flere elementer som spiller inn: Har noden ventende pakker som skal sendes over denne linken? Hvis ikke, vil det i så fall på det tidspunktet denne linken inntreffer være på tide å generere og sende en keep-alive-pakke til linkens nabo, eller eventuelt kringkaste en advertise-pakke? Hvis svaret på noen av disse spørsmålene evalueres til ja er linken en kandidat til behandling og tas med videre. Vi beregner hvilken ASN linken skal opptre på, mellomlagrer ASN i link-strukturen, og legger linken på en kandidatliste som er sortert etter denne verdien. Dermed sitter vi til slutt igjen med en sortert liste over linker hvor listens lengde er lik antallet aktive superrammer. Algoritmen, som kjøres som første del av hver linkfordeling, er vist i pseudokode nedenfor:

```
foreach (superframe in superframes)
  if (superframe->active)
     // Hent den neste linken som inntreffer hos superrammen:
     foreach (link in superframe->links)
      if (link == RX || link->waiting_packets != NULL)
        * Beregn ASN linken inntreffer på
        * Legg kandidaten på linklista, med plassering basert på ASN
        break;

      else if (link == TX) // Sendelink _uten_ ventende pakker
        * Beregn ASN linken inntreffer på.
        * Hvis på tide med utsending av advertise eller keep-
          alive på beregnet ASN:
          * Legg kandidaten på linklista, med plassering basert på ASN
        break;
```

Siden alle linkene er «behandlingsbare» og sorterte skal vi nå i utgangspunktet kunne sette i gang med behandling av den første, men dette kompliseres i tilfeller hvor en eller flere av de etterfølgende linkene skal inntreffe på samme ASN. I slike tilfeller skal linkene prioriteres etter følgende kriterier:

- Mottakslinker prioriteres etter superramme-ID. Lavest først.
- Sendelinker prioriteres før mottakslinker
- Sendelinker prioriteres etter ventende pakkes prioritet, dernest etter hvor lenge det er siden det ble kommunisert med naboen i andre enden av linken

Det kan være verdt å bemerke at det innenfor spesifikasjonen, med disse nevnte kriteriene, ligger en viss mulighet for at trafikken på en link alltid blir overkjørt av trafikken på en annen link. Dette forutsetter at strukturen med superrammer og linker er lagt opp slik at to linker hele tiden inntreffer samtidig, og at trafikken på den ene linken alltid har høyere prioritet enn den andre. Dette er imidlertid ikke spesielt sannsynlig siden all normal trafikk foregår med likt prioritetsnivå og link dermed velges ut fra hvor lang tid det er siden forrige kommunikasjon i stedet. I tillegg kan en slik situasjon motarbeides ved at nettverksadministratoren sørger for en god superramme- og linkstruktur hvor ikke to linker alltid inntreffer i samme tidslomme.
Vi har implementert prioriteringen noenlunde slik pseudokoden på neste side viser:

```
boolean first = true;

foreach (link in links)
{
  if (links.size < 2) // Ingen vits i å prioritere én link
    break;

  if (first == true)
    first = false;
    * sett link som 'candidate' og fortsett med neste link

  else {
    * Sett link som 'challenger'

    if (candidate->asn == challenger->asn) {
      if (begge er RX)
        * sett linken tilhørende superramme med lavest ID som 'candidate'

      else if (begge er TX)
        * sett linken med høyest prioritert pakke som 'candidate'
        if (candidate->priority == challenger->priority)
          * undersøk hvor lenge det er siden forrige kommunikasjon med
            deres respektive naboer
          * sett den med høyest tidsforskjell som 'candidate'

      else if (en RX og en TX)
        * sett TX-linken som 'candidate'

      else
        * Planlegg 'candidate' til å slå til på sin tilknyttede ASN *
        return; // linkfordeling utført
    }
  }
} // foreach end

// Vi hadde bare en link i listen vår:
* Planlegg linken til å slå til på sin tilknyttede ASN
```

I de aller fleste tilfeller vil linkene ha ulike tiltenkte ASN-er, og derfor vil det normalt holde å sammenligne nummer én og to, for så å planlegge den første av dem.

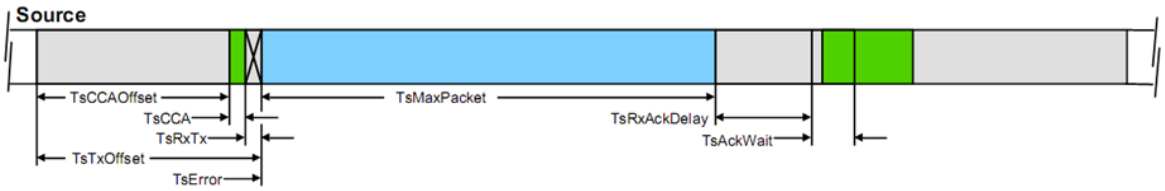
Vi ser at vi avslutter algoritmen med å planlegge én link til å slå til på sin tilknyttede ASN. For å oppnå dette benytter vi oss av AVR2025-bibliotekets tidsstyringsmekanismer. AVR2025 lar oss legge et funksjonskall på vent, og sørger for å utføre kallet på et gitt tidspunkt eller etter en gitt tid. I vårt tilfelle kan vi enkelt beregne tidspunktet ved å multiplisere linkens tiltenkte ASN med tidslommenes lengde, 10000 mikrosekunder (= 10 ms), og vi har nå tidspunktet linken skal behandles på oppgitt i mikrosekunder. Dermed overlater vi til AVR2025s tidshåndterer å kalle på vår linkbehandler når dette tidspunktet inntreffer. Protokollen legger, som vi snart skal se, opp til en viss behandlingstid i tidslommen før selve sendingen eller mottaket av data starter, så det er ikke nødvendig å legge inn slakk i kallet på linkbehandleren. Alt av for- og etterarbeid (pakkekonstruksjon, køhåndtering, ny linkplanlegging etc.) skal kunne utføres innenfor tidslommens ti millisekunder. Det bør bemerkes at vi i pseudokoden for linkfordeleren foreslått ovenfor, og i sendemotoren som omtales nedenfor, ikke har lagt inn behandling av delte linker. For å implementere dette må det legges til en if-test for tilfeller der linken som behandles er en delt sendelink, og i så fall må CSMA/CA-algoritmen utføres som omtalt i kapittel 4.3.5.

Linkbehandleren

Ved tidspunktet for en planlagt tidslomme skal en link behandles, og dette håndteres av linkbehandleren (i vår implementering en funksjon kalt `serve_timeslot_cb`), som videreformidler håndteringen til enten sendemotoren eller mottaksmotoren. Linkbehandleren avgjør om det dreier seg om en sende- eller mottakslink, og i tilfellet sendelink blir sendemotoren kalt, og i tilfellet mottakslink mottaksmotoren kalt. Vi skal nå se nærmere på disse to modulene.

Sendemotoren

Sendemotoren, kalt XMIT-motoren i WirelessHART-standarden, tar seg av selve utsendingen av en DLPDU, og i vårt tilfelle også mottak av eventuell påfølgende ACK. Underveis i denne prosessen skal den holde styr på timing i henhold til Figur 6-8. Figuren er hentet fra standarden, og viser de ulike hendelsene innenfor en tidslomme på avsendersiden. Grønn farge betyr at radioen lytter, blå farge betyr at radioen sender, og grå farge er tiden der radioen ikke benyttes og kan holdes avslått. Vi skal her se nærmere på de ulike punktene i denne figuren, og se hvordan de er relatert til implementeringen og de ulike tjenesteprimitivene. De ulike tidsverdiene i denne figuren er tidligere omtalt i kapittel 4.3.5.



Figur 6-8 Timing ved sending av pakke

Vi minner om at vi i vår implementering utfører alle hendelser sekvensielt og i én prosess/tråd, og tiden det tar å for eksempel sende en pakke benyttes derfor ikke til noe annet enn å vente på at det fysiske laget skal bekrefte at overføringen er ferdig. Alle hendelser, små og store, foregår én om gangen og etter tur.

Overføringen av en DLPDU starter ved at tidspunktet for en planlagt tidslomme med en sendelink med ventende pakke inntreffer, og tidshåndtereren i AVR2025 kaller på linkbehandleren som videreformidler kallet til sendemotoren. Sendemotoren setter i gang med å få generert en datapakke av den køede datastrukturen gjennom LLC-laget. På denne pakken blir det i vår implementering også lagt på en PHY-header slik at en ferdig dataramme er klar til sending over linken. Videre klargjøres radioen til mottak for utføring av CCA (Clear Channel Assessment) med kallet på funksjonen for ENABLE.request-primitivet. Alt dette må være ferdig utført innen tiden `TsCCAOffset` fra tidslommens start har passert. Ved tidspunktet `TsCCAOffset` fra tidslommestart utføres så CCA, og radioen klargjøres deretter til sending med et nytt kall til ENABLE.request-funksjonen. På sendetidspunktet, `TsTxOffset` etter starten av tidslommen, får det fysiske laget beskjed om at det skal sende rammen ved hjelp av kallet på funksjonen for DATA.request-primitivet. Hele denne flyten i tjenesteprimitiver er illustrert i Figur 6-9. På neste side har vi foreslått en algoritme for sendemotoren i pseudokode. Denne algoritmen er ikke helt identisk med vår egen, da vår implementering av sendemotoren ikke er fullstendig og vi blant annet ikke ennå har inkludert CCA i våre dataoverføringer. Alle tjenesteprimitivene, CCA inkludert, skal imidlertid være implementert og tilrettelagt for å kunne utføres etter denne algoritmen:

```
* kalkuler tidslommens starttidspunkt:
      slot_start_us = current_absloute_slot_number * 10000;
* endre tilstand i TDMA-maskin til «Talk»

frame = frame_create(transmit_packet_queue_entry);

* kalkuler kanal det skal sendes på basert på tidslomme og
  linkens kanaloffsett

if(!CCAEnabled)
      TxDelay = slot_start_us + TsTxOffset;
else {
      enable_request(RX, channel);
      * Hvis confirm:
      TxDelay = slot_start_us + TsCCAOffset;
      wait_until(TxDelay); // Form for spinlock
      cca_request();
      * Hvis confirm:
      TxDelay = TxDelay + TsCCA + TsRxTx;
}
enable_request(TX, channel);
* Hvis confirm:
wait_until(TxDelay);
data_request(frame);
* Hvis confirm:
* endre tilstand til «Idle» hvis pakken ble kringkastet, ellers endre
  tilstand til «Wait for ACK»
```

Så langt har vi fått sendt pakken av gårde, men i en normal overføring fra én nabo til en annen vil vi også forvente en ACK tilbake. ACK-mottaking tilhører i følge standarden mottaksmotoren, og dette virker logisk siden en ACK i realiteten er en datapakke som skal mottas. Mottak av ACK er imidlertid en viktig del av en vellykket normal pakkesending og skjer i samme tidslomme som utsendingen. Derfor mener vi at det også gir mening å tilordne dette til sendemotoren. Siden vi mener den siste varianten blir ryddigst, har vi valgt denne i vårt forslag til implementering.

Ved ferdig utsendt pakke, altså etter at det avsluttende kallet på `data_request(frame)` i algoritmen over er bekreftet med en tilsvarende `data_confirm(frame)`, befinner vi oss

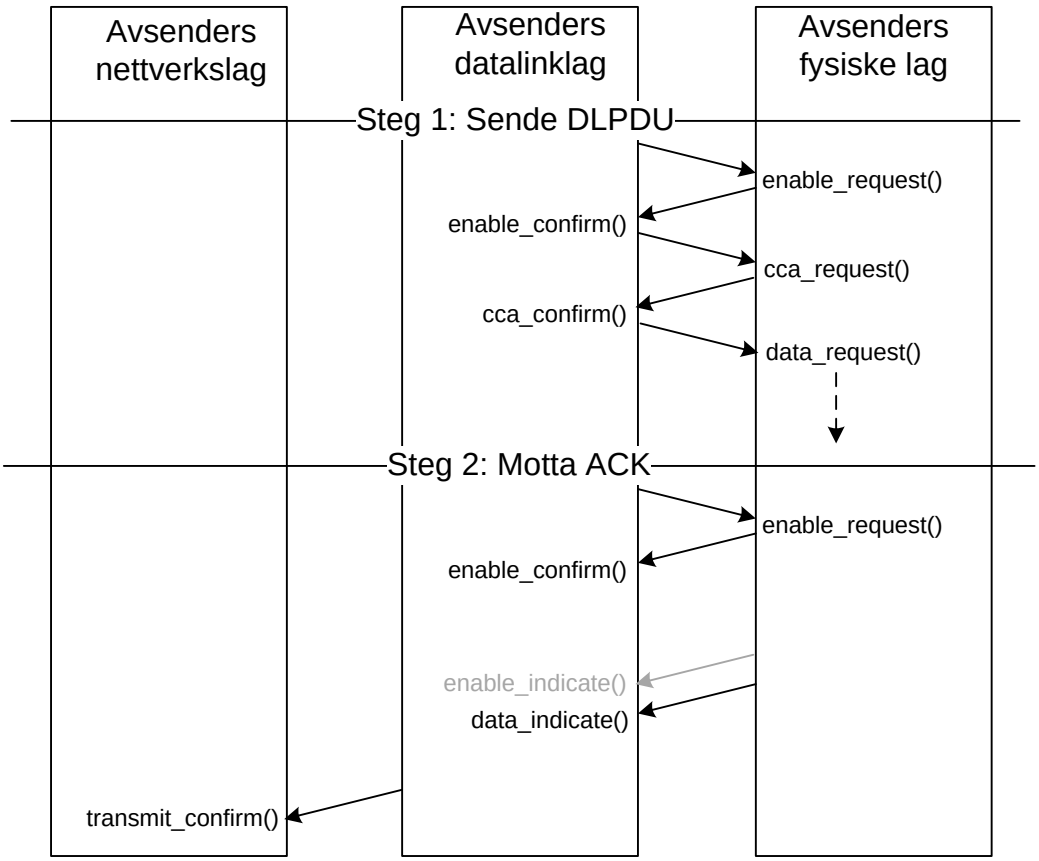
på slutten av den blå rektangelen i Figur 6-8 og TDMA-tilstandsmaskinen står i tilstanden «Wait for ACK». Med andre ord har tiden `TxDelay` (se pseudokode over) pluss pakkeoverføringstiden for den gitte pakken forløpt siden tidslommens start. Siden mottakeren må få tid til å tolke den mottatte pakken og generere en ACK-DLPDU for sending tilbake, er det i spesifikasjonen lagt inn en pause av lengde `TsRxAckDelay` før det er nødvendig å snu radioen og begynne å lytte på mediet. I vår implementering snus radioen umiddelbart, men dette har ingen videre praktisk betydning utenom muligens noe høyere energiforbruk fordi radioen da beholdes påslått i tiden mellom avsluttet sending og starten på mottak av ACK. I en komplett implementering bør det vurderes å slå radioen av i denne perioden. Ved mottatt ACK med responskode om at alt er i orden, kan den avsendte pakken fjernes fra den utgående pakkekøen. Hvis ACK ikke mottas innenfor perioden `TsAckWait` etter at vi skal begynne å forvente den, derimot, eller responskoden ga negativ tilbakemelding, lar vi pakken forbli på køen. Eventuell retransmisjon eller fjerning av pakke grunnet timeout overlates til linkfordeleren. Uansett er sendemotoren nå ferdig med jobben sin, og avslutter med å kalle på linkfordeleren på nytt for å planlegge neste link som skal behandles. Følgende algoritme foreslås for ACK-delen av sendemotoren (pseudo):

```
enable_request(Rx, Channel);
// Ved confirm av ENABLE.request er vi klare til mottak, og følgende
skjer:
* legger en ack-behandlingsfunksjon på timeout som inntreffer etter
  TsRxAckDelay + TsAckWait + overføringstid for ACK, og som sørger for:
      * sjekke om ACK er mottatt ved å se på en statusvariabel satt av
        data_indicate()-funksjonen initiert fra PHY ved mottak
      * validere eventuelt mottatt ACK
      * fjerne den avsendte pakken fra utgående kø hvis ACK er OK
      transmit_confirm(...);
      * i alle tilfeller:
      link_scheduler();
```

Vi ser av algoritmen at radioen i vårt tilfelle beholdes på til vi er sikre på at hele den eventuelle ACK-en er mottatt. I følge spesifikasjonen skal vi kunne slå av radioen etter tiden `TsRxAckDelay` + TsAckWait hvis vi ikke har oppdaget at data er i ferd med å komme inn på radioen. Med andre ord at vi ikke har mottatt en ENABLE.indicate-primitiv fra det fysiske laget innenfor denne tiden. Siden vi i vår implementering ikke har funnet noen måte å

oppdage dette på før overføringen er ferdig, må vi imidlertid beholde radioen påslått helt til vi er sikre på at en eventuell ACK er mottatt, altså `TsRxAckDelay + TsAckWait +` ACK-overføringstid. Dette betyr noe høyere energiforbruk i de tilfellene hvor ACK ikke blir mottatt, men dette vil forhåpentligvis bare være unntaksvis.

Hvis vi trekker sendemotorfunksjonaliteten ut fra Figur 6-4 på side 107 får vi Figur 6-9. Her er det kun tjenesteprimitivene som er tegnet inn, og ikke den mellomliggende logikken. ENABLE.indicate-funksjonen er igjen grånet ut, siden den er implementert som en del av programflyten men i realiteten ikke har noen praktisk funksjon i vår implementering.

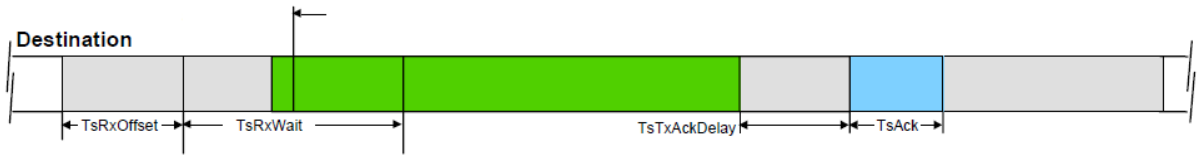


Figur 6-9 Gangen i avsenders datalinklag og fysiske lag i sendemotoren

Mottaksmotoren

Mottaksmotoren, kalt RECV-motoren i WirelessHART-standarden, tar seg av mottaket av en DLPDU i en mottakslink, samt sørger for at ACK blir generert og sendt tilbake til avsender i de situasjoner der dette er påkrevet. Underveis i denne prosessen skal den holde styr på timing i henhold til Figur 6-10, som viser de ulike hendelsene innenfor en tidslomme på mottakersiden. Også her betyr grønt at radioen lytter og blått at radioen sender. Vi skal, som i

avsnittet om sendemotoren ovenfor, se nærmere på de ulike punktene i figuren, og se hvordan de er relatert til implementeringen og de ulike tjenesteprimitivene. For detaljene om tidsverdiene i figuren refererer vi nok en gang til side 67 i kapittel 4.3.6.



Figur 6-10 Timing ved mottak av pakke

Mottak av DLPDU starter ved at tiden for en tidslomme med en planlagt mottakslink inntreffer, og mottaksmotoren klargjør radioen i mottaksmodus med kallet på funksjonen for ENABLE.request-primitivet. Lyttingen skal starte etter `TsRxOffset`, men pakken skal ideelt sett ikke begynne å komme inn før midtveis i den neste tidsperioden, `TsRxWait`. Grunnen til at lyttingen starter tidligere enn forventet mottakstidspunkt er for å tillate en viss slingring i tidssynkroniseringen. Etter tiden `TsRxOffset` + `TsRxWait` etter tidslommestart har passert, skal vi i følge spesifikasjonen kunne slå av radioen hvis vi ikke har fått indikasjoner fra det fysiske laget om at en pakke er på vei inn. Siden vi, som tidligere nevnt, ikke har funnet noen måte for radioen å varsle det fysiske laget om dette på før hele pakken er mottatt, beholder vi radioen på til `TsRxOffset + TsRxWait +` maksimal pakkeoverføringstid har passert etter tidslommens start, med mindre det fysiske laget gir indikasjon om at en pakke er ferdig overført før den tid. Deretter kommer en tidsperiode på `TsTxAckDelay` der vi kan slå av radioen. Hvis vi ikke fikk noe data inn i lytteperioden er mottaksmotoren ferdig, og vi kan nå kalle linkfordeleren med en gang. Dette skjer normalt fordi naboen tilhørende linken ikke hadde data å sende i denne omgang. Mottok vi data benyttes perioden til å analysere og validere pakken som kom inn, og hvis alt er i orden konvertere dette bufferet til en datastruktur som egner seg for køing og raske oppslag, samt legge denne strukturen på køen for innkommende data hvis pakken ble korrekt mottatt og var adressert direkte til oss. Deretter genereres ACK med informasjon om tidsavvik og hvorvidt det var plass til pakken på mottakskøen, slik kapittel 4.3.4 beskriver. Denne ACK-en sendes tilbake til avsender i det `TsTxAckDelay` har utløpt, og etter at overføringen av ACK er ferdig kalles linkfordeleren for planlegging av neste link som skal behandles. Hele det grå feltet mot slutten av figuren (etter `TsAck`) er avsatt til denne planleggingen.

Linkfordeleren spiller en viktig rolle i MAC-sublaget og eksekveringen av den kan risikeres å bruke mye tid. I tilfeller hvor det er planlagt en utlink i tidslommen som kommer umiddelbart etter en tidslomme hvor noden har mottatt en datapakke er det viktig at linkfordeleren rekker å utføre oppgavene sine i tide. Vi har ikke hatt anledning til å måle tidsbruken i linkfordeleren siden den ikke er fullstendig implementert, men vi kan regne oss frem til tiden som gjenstår i en tidslomme til dette formålet. For å finne tiden som er til overs når det overføres en DLPDU av maksimal lengde tar vi for en mottakernode tidslommens lengde (10 millisekunder, eller 10 000 mikrosekunder) og trekker fra summen av tidsvariablene TsRxOffset, TsRxWait, TsMaxPacket, TsAckDelay og TsAck. For en avsendernode tar vi tidslommens lengde og trekker fra summen av TsTxOffset, TsMaxPacket, TsRxAckDelay, TsAckWait og TsAck. Vi ser på to slike tilfeller (verdiene i regnestykket er hentet fra Tabell 4-3 på side 55):

- *DLPDU-en kommer inn like før vinduet for å motta pakker lukkes.*
  Regnestykket for en avsender blir:
  10 000 us – (2120 + 4256 + 800 + 400 + 832) us = 1592 us
  Regnestykket for en mottaker blir:
  10 000 us – (1120 + 2200 + 4256 + 1000 + 832) us = 592 us
- *DLPDU-en kommer inn på forventet tidspunkt*
  Regnestykket for en avsender blir likt som over, siden tidspunktet for utsending av en pakke ikke kjenner vinduet, men i stedet forholder seg til et gitt tidspunkt.
  Regnestykket for en mottaker blir:
  10 000 us – (1120 + 1100 + 4256 + 1000 + 832) us = 1692 us

Vi ser at tiden en mottaker har til rådighet før en eventuell planlagt utsending i en umiddelbart etterfølgende tidslomme er knapp. I kapittel 7.3 skal vi se at en node bruker 493 mikrosekunder på å evaluere den innkommede datapakken samt lage en ACK-pakke, og da vi antar at linkplaneleggeren har behov for mer enn 592 mikrosekunder (som «worst case»-regnestykket gir oss) for å utføre oppgavene sine tror vi en sending umiddelbart etter et mottak kan bli vanskelig å gjennomføre ved maksimal pakkelengde og maksimal slingring i tidssynkroniseringen. Kommer derimot pakken inn på det tidspunktet man forventer en pakke vil nok tidsplanleggeren ha nok tid (1692 mikrosekunder). Hvis mulig bør nettverksadministratoren settes opp slik at utlinker ikke forekommer i tidslommer som er ummidelbart etter en tidslomme med en innlink. Uansett er dette scenarioet maksimalt

uheldig, og forhåpentligvis vil det være ytterst sjelden at to tidslommer opptrer umiddelbart etter hverandre, samtidig som avviket i synkroniseringen er maksimal og pakken som overføres er maksimal. I slike tilfeller risikerer vi å måtte stole på at en retransmisjon rydder opp.

Nedenfor har vi foreslått en algoritme for mottaksmotoren i pseudokode. Også her skiller algoritmen seg noe fra hva vi har rukket å implementere selv, og er derfor snarere en representasjon av designet vi har jobbet etter enn av selve implementeringen.

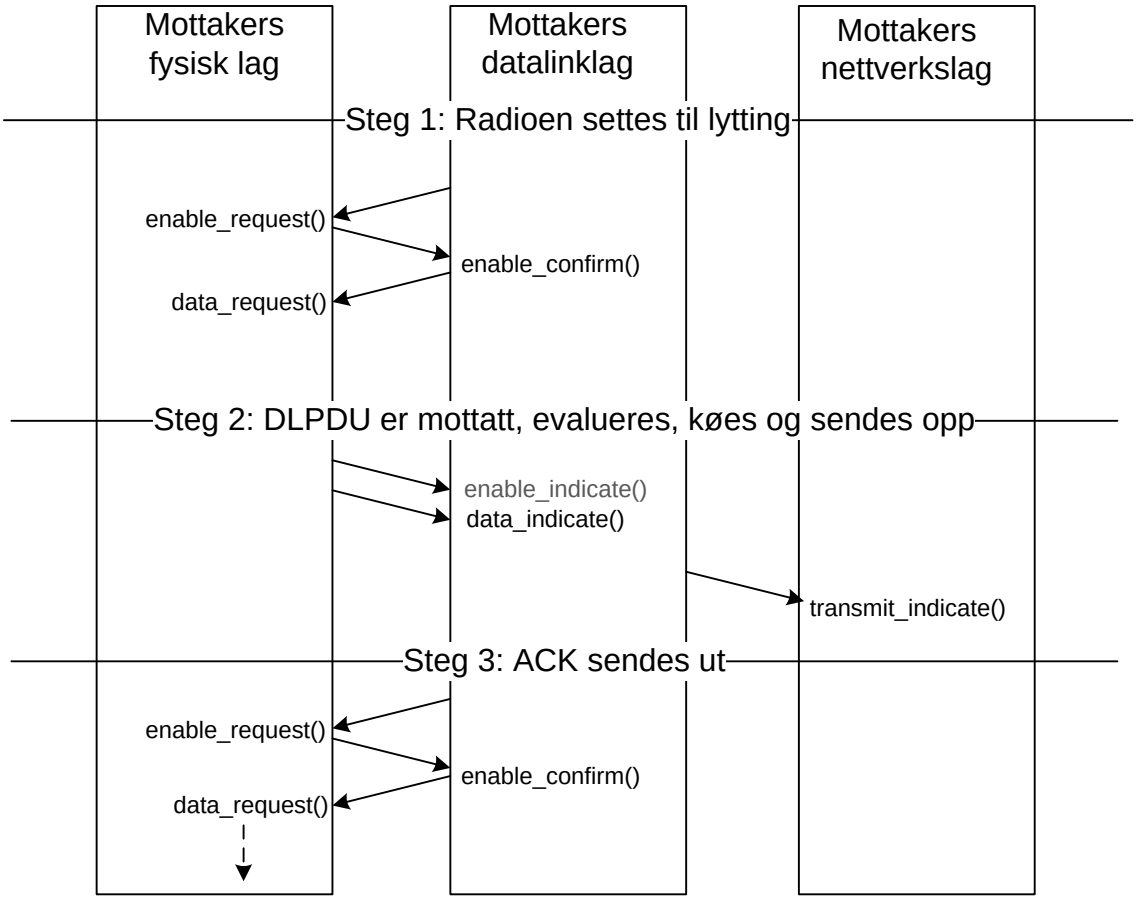
```
* kalkuler tidslommens starttidspunkt:
      slot_start_us = current_absloute_slot_number * 10000;
* endre tilstand i TDMA-maskin til «Listen»
* kalkuler kanal det skal lyttes på basert på tidslommenr og
  linkens kanaloffset
* beregn tidspunkt for å slå på radioen:
      RxDelay = slot_start_us + TsRxOffset – slack_for_enable_request;
      wait_until(RxDelay);
      enable_request(RX, channel); // Slå på radio
* Hvis confirm:
      data_request(ready_buffer);
* Hvis confirm:
      * Legg en funksjon på timeout som slår til etter RxDelay + TsRxWait
        + maksimal pakkeoverføringstid, og som har ansvar for å
            * slå av radioen
            * endre tilstand til «Idle»
            * kalle på linkfordeleren
```

Algoritmen hittil har ingen behandling av eventuelt mottatt pakke. Den slår bare av radioen etter maksimal overføringstid. Vi må derfor utvide mottaksmotoren med en pakkebehandler. Denne initieres fra det fysiske laget ved et kall på `data_indicate(data)`, som forteller at en pakke er ferdig mottatt. Denne delen av mottaksmotoren foreslås implementert slik:

```
// data_indicate-funksjonen:

* Avbryt funksjonen lagt på timeout i mottaksmotoren. Den skal bare slå
  til hvis data_indicate ikke inntreffer innen den spesifiserte tiden

if (is_adressed_to_me(frame) == false) {
      * oppdater kommunikasjonsstatistikk
} else {
      if(check_crc(frame) != OK)
            * oppdater kommunikasjonsstatistikk
      else if(check_mic(frame) != OK)
            * oppdater kommunikasjonsstatistikk
            * oppdater sikkerhetsstatistikk (siden feil i MIC kan tyde på
              angrep på nettverket)
      else
            * kalkuler TsError (tid mellom når pakken var forventet inn og
              når den faktisk kom inn på radioen)
            * konverter mottatt pakke fra et buffer til en datastruktur
            * legg på mottakskøen hvis plass
            * generer og send ut ACK (hvis ikke kringkastingspakke) med
              info om TsError og hvorvidt pakken fikk plass på køen
            * oppdater statistikk for nabo som sendte pakken
            * send TRANSMIT.indicate til NW-laget
}

* slå av radioen

TDMA_state = TDMA_IDLE;
link_scheduler();
```

Legg merke til at vi her har underlagt ACK-genereringen til mottaksmotoren, på samme måte som vi tidligere har lagt ACK-mottak til sendemotoren. Dette er, som tidligere omtalt, ulikt designet i standarden, men gir etter vår mening en mer sammenhengende implementering.

Vi kan, på samme måte som vi gjorde for sendemotorfunksjonaliteten, trekke ut tjenesteprimitivflyten som utgjør mottaksmotoren fra Figur 6-4. Dette gir oss figur Figur 6-11 under. Merk at her er det kun tjenesteprimitivene som er tegnet inn, og ikke den mellomliggende logikken.



Figur 6-11 Gangen i avsenders datalinklag og fysiske lag i sendemotoren

7 Evaluering

I dette kapitlet ser vi nærmere på hvordan noen av egenskapene vi har implementert faktisk fungerer. I kapittel 6 så vi at vi har utviklet en implementering av det fysiske laget og at vi også har store deler av en implementering av datalinklaget på plass. Protokollen slik vi har implementert den er fremdeles i «startgropen» og egenskapene som lar seg teste er nokså basale. Vi mener dog at implementeringen har kommet et godt stykke i løpet av de ni månedene vi har hatt til rådighet.

Vi kommer til å se på pakkeutsending, trafikken mellom to noder, inspisere pakkestrukturen under overføringen, hvordan utsending av ACK fungerer og så vidt litt om tidsstyring samt hvordan enkel kommunikasjon mellom noder fungerer når tre noder er del av nettverket, for å få en bekreftelse på at en node gjenkjenner pakker som er adressert til seg.

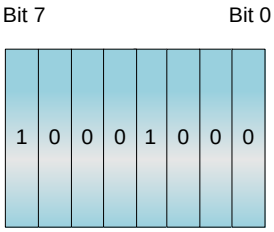
Vi har benyttet pakkesnifferen (beskrevet i kapittel 5.3) og medfølgende programvare til å overvåke og visualisere trafikken.

7.1 Pakkeutsending

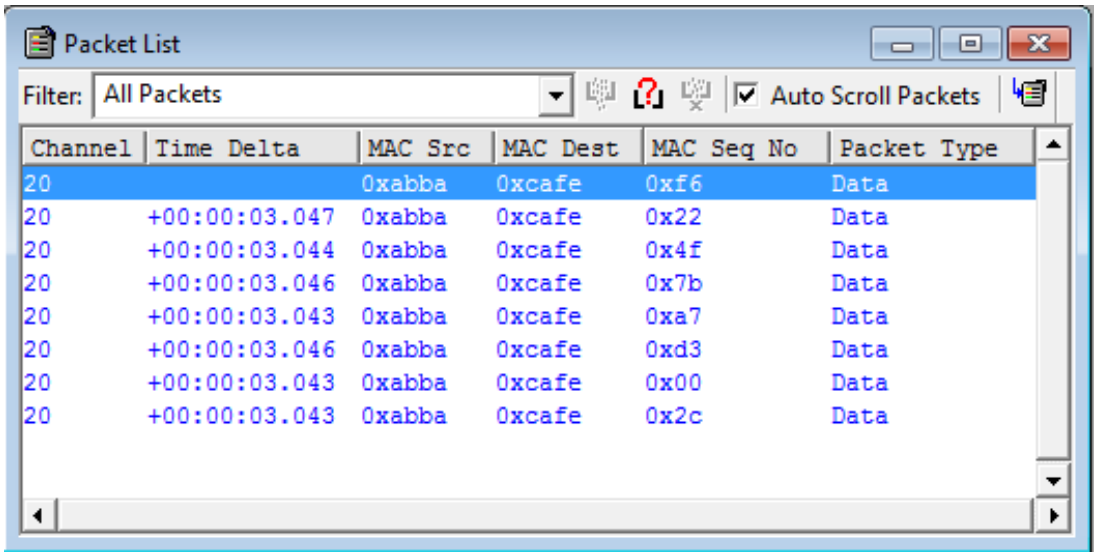
Først skal vi se på hvordan pakkene ser ut når de sendes ut fra radioen på en enkelt node, og vi vil samtidig kontrollere at adresseringen fungerer som den skal. Vi så i kapittel 0 at WirelessHART-pakker har et kontrollfelt som korresponderer med andre IEEE 802.15.4-pakker og at snifferen dermed ikke har noen problemer med tolking av overføringsdata. Det er derfor naturlig å se på adressering og visualisering av datatrafikken som pakkesnifferen plukker opp når vi vil undersøke den enkleste trafikken fra en node.

Vi programmerer en node med kortadresse $ABBA_{16}$. DLPDU-en noden sender ut er addressert til $CAFE_{16}$ og DLPDU-adressespesifikatoren er bygget opp til å beskrive at begge nodene benytter sine korte adresser. (Se kapittel 4.3.4 for mer detaljer om oppbygningen av en DLPDU). Det er bare bitene to og seks i adressespesifikatoren som WirelessHART-standarden tillater at endres, og med kort adresse for både sender (bit nummer to satt til 0) og mottaker (bit nummer seks satt til 0) blir adressespesifikatoren som vist i Figur 7-1, og med den ledende byten på enhver WirelessHART-pakke satt til 41_{16} og når vi vet at felter i pakkehodet og –halen sendes minst signifikant byte først (beskrevet i kapittel 4.3.4) kan vi forvente å se 88 41_{16} som rammekontrollfelt i pakken når pakkesnifferen plukker den opp. Dette fordi pakkesnifferen forventer at de to første bytene er ett felt, slik de er i IEEE

802.15.4, og dermed bytter rekkefølgen på dem for presentasjonen til brukeren. I realiteten overføres dataene som feltene 41_{16} og 88_{16} i WirelessHART.



Figur 7-1 Adressespesifikator som brukt i evalueringen



Figur 7-2 Trafikk plukket opp av pakkesniffer når en node sender

Figur 7-2 viser pakker som pakkesnifferen har oppdaget, og kolonnene som vises beskriver hvilken kanal trafikken er oppdaget på, tiden siden forrige pakke i millisekunder, avsenderadresse, mottakeradresse, sekvensnummer og type pakke. Type pakke vil for all WirelessHART-trafikk være data da den ledende 41_{16} tolkes som datapakke i IEEE 802.15.4-sammenheng.

Vi ser at kanalen er kanal 20, noe som stemmer overens med vår benyttelse av tjenesteprimitivfunksjonskallet `enable_request(Talk, 20)` som angir retning og kanal før utsending av data. Den neste kolonnen, «Time Delta», viser tidsdifferansen mellom pakkene som pakkesnifferen registrerer. Denne verdien er i millisekunder og noden er for anledningen programmert til å sende ut en pakke (om det ligger noen på køen) hvert tredje sekund. Vi ser at tidsdifferansen er tre sekunder pluss en verdi mellom 43 og 47

millisekunder. Vi er ikke sikre på hva disse ekstra millisekundene skyldes, men antar at årsaken kan være knyttet til hardware i pakkesnifferen eller hvordan programvaren til pakkesnifferen forholder seg til granulering av tid. En annen sannsynlig årsak er små forskjeller i krystallene, som kan løses ved hjelp av å kalibrere den interne RC-oscillatoren [51]. Når det gjelder adresseringen ser vi at både avsender og mottaker tolkes slik vi programmerte: Avsender er $ABBA_{16}$ og mottaker er $CAFE_{16}$.

Kolonnen «MAC Seq No» viser sekvensnummeret slik enheten har lagt til pakken og vi husker fra kapittel 4.3.4 at sekvensnummeret består av den minst signifikante byten i ASN (Absolute Slot Number). Om vi ser på to pakker som er sendt etter hverandre kan vi regne ut om sekvensnummeret stemmer ved å ta tidsdifferansen mellom de to pakkene (gitt ved kolonnen «Time Delta» i figuren) og se om sekvensnummeret til pakke nummer to korresponderer med denne tidsdifferansen: Med en tidsdifferanse på 3000 millisekunder vil det være 300 tidslommer mellom hver utsending og ASN vil også øke med 300. På en byte er det kun plass til å beskrive 256 tidslommer og da sekvensnummeret kun skal være én byte vil sekvensnummeret telle rundt seg selv for hver utsending. La oss ta for oss de to første pakkene i Figur 7-2:

$$Sekvensnummer1 = F6_{16}$$

$$Sekvensnummer2 = 22_{16}$$

$$Antall\ tidslommer\ mellom\ pakkene = 12C_{16}\ (300_{10})$$

$$F6_{16} + 12C_{16} = 222_{16}$$

$$Minst\ signifikante\ byte\ av\ 222_{16} = 22_{16}$$

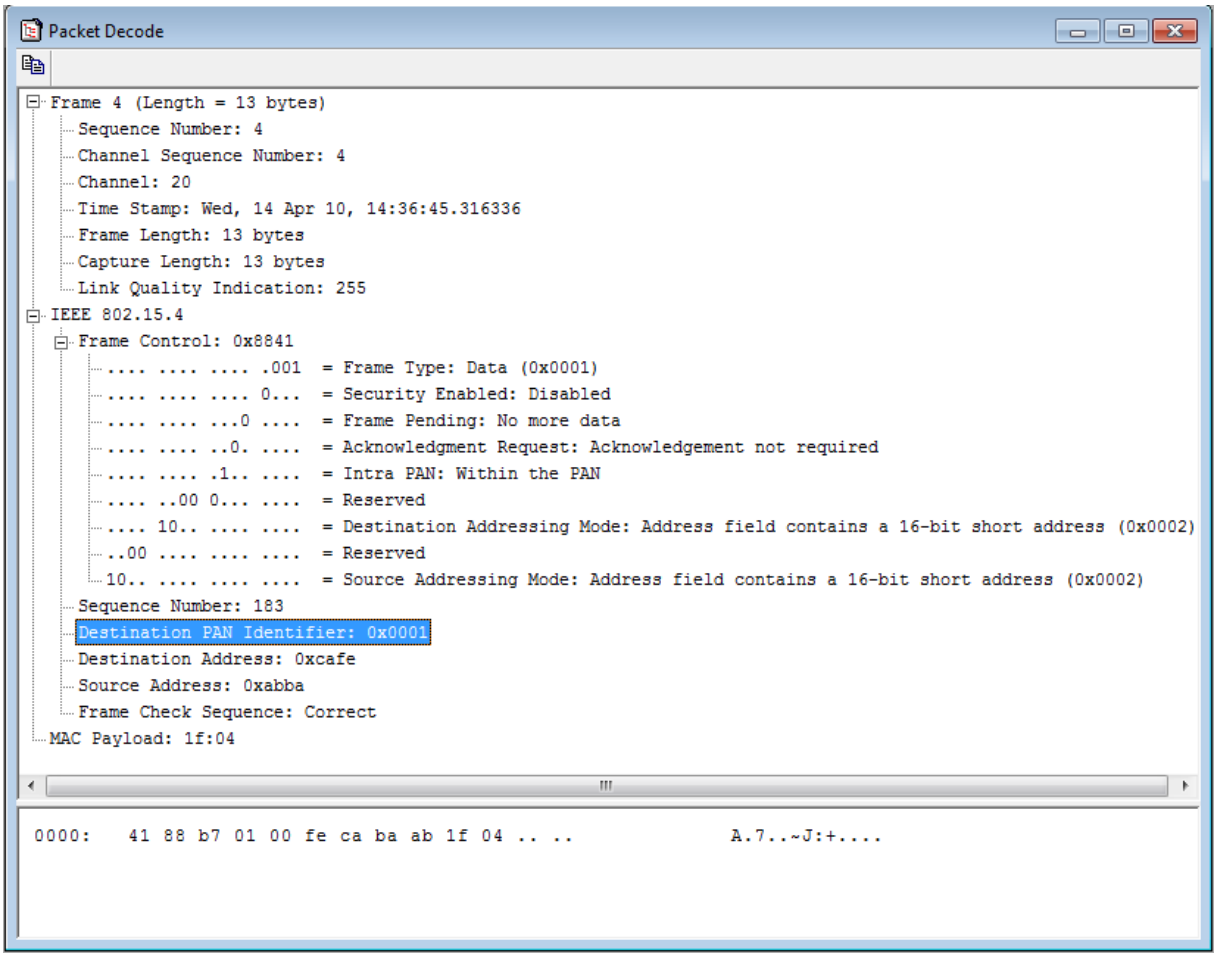
Vi ser altså at sekvensnummeret til den andre pakken (22_{16}) stemmer overens med at noden sender ut en pakke hvert 3000 millisekund.

7.2 Pakkestrukturen på nett

Vi behandler strukturen til en WirelessHART-pakke (DLPDU) i kapittel 4.3.4 og vi har beskrevet hvordan vi konstruerer den i vår implementering i kapittel 6.4.2. Vi skal i dette kapitlet se nærmere på om vår implementering tilfredsstiller kravene fra spesifikasjonen. Når vi tester den faktiske strukturen benytter vi oss igjen av pakkesnifferen og ser på pakkene slik de fremstår i selve overføringen. Vi så i evalueringen over hvordan DLPDU-adressespesifikatoren sammen med verdien 41_{16} fremsto når pakkene ble plukket opp av

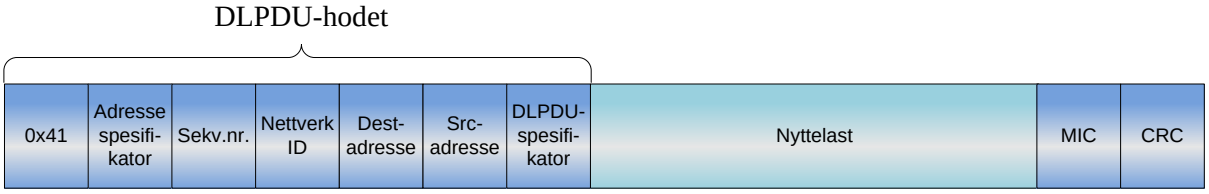
pakkesnifferen, og vi er her interessert i å se på de resterende feltene i DLPDU-hodet og innholdet i nyttelasten for å forsikre oss om at pakkene bygges opp korrekt.

Figur 7-3 viser pakkestrukturen slik pakkesnifferen tolker den, og vi skal konsentrere oss om informasjonen som er samlet under «IEEE 802.15.4» i figuren. Det første som vises er en beskrivelse av hver bit i rammekontrollfeltet («Frame Control») og vi ser at blant annet 41_{16} beskriver rammen som datatype, men som vi har vært inne på tidligere sendes alle WirelessHART-pakker med en ledende byte med verdien 41_{16}, noe som gjør at man kan i WirelessHART-sammenheng se bort fra tolkningen figuren fremstiller. Vi ser videre at adressespesifikatoren (88_{16}) definerer begge adressatene til å benytte sine korte adresser. Deretter beskrives sekvensnummer ($183_{10,}$ $B7_{16}$), mottaker- og avsenderadresse og en validering av CRC. CRC-en blir validert av programvaren (Sensor Network Analyzer) til pakkesnifferen og den faktiske verdien blir ikke vist frem. Til slutt vises det som programvaren tolker som nyttelasten. Den første byten i denne nyttelasten er i realiteten det siste feltet i hodet til WirelessHART-DLPDU-en: DLPDU-spesifikatoren, mens alt etter denne er nyttelasten. Siden vi ikke har implementert sikkerhet mangler MIC-feltet, som normalt ville kommet etter nyttelasten.

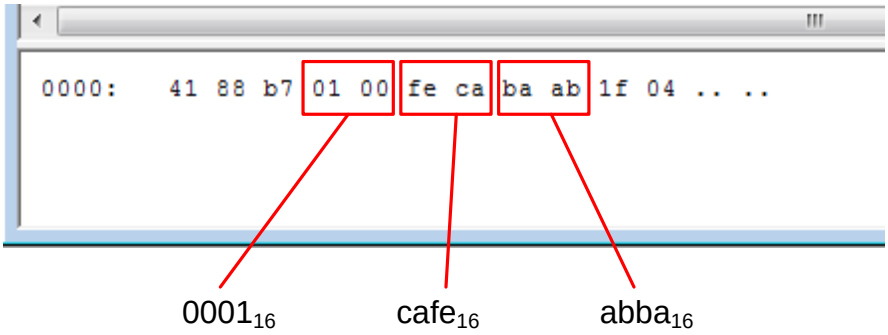


Figur 7-3 Pakkestruktur

Figur 7-4 (som vi også har sett som Figur 4-4 tidligere i rapporten), viser DLPDU-strukturen, og Figur 7-5 viser disse feltene slik de plukkes opp av pakkesnifferen.

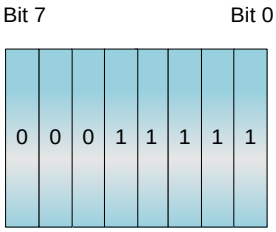


Figur 7-4 DLPDU-struktur



Figur 7-5 Feltene i pakken sendes minst signifikant byte først

Vi kjenner igjen 41_{16}, adressespesifikatoren (88_{16}) og sekvensnummeret ($b7_{16}$). Deretter kommer det tre felt som hver består av to byte, og som beskrevet i kapittel 4.3.4 blir alle felt i pakkehodet som er større enn én byte sendt minst signifikant byte først. Den neste byten ($1f_{16}$) er DLPDU-spesifikatoren, før nyttelasten er det siste som plukkes opp. Figur 7-6 viser DLPDU-spesifikatoren slik den ble plukket opp av pakkesnifferen ($1f_{16}$). (Detaljert beskrivelse om oppbygging av DLPDU-spesifikatoren vises blant annet i Figur 4-6 på side 49).

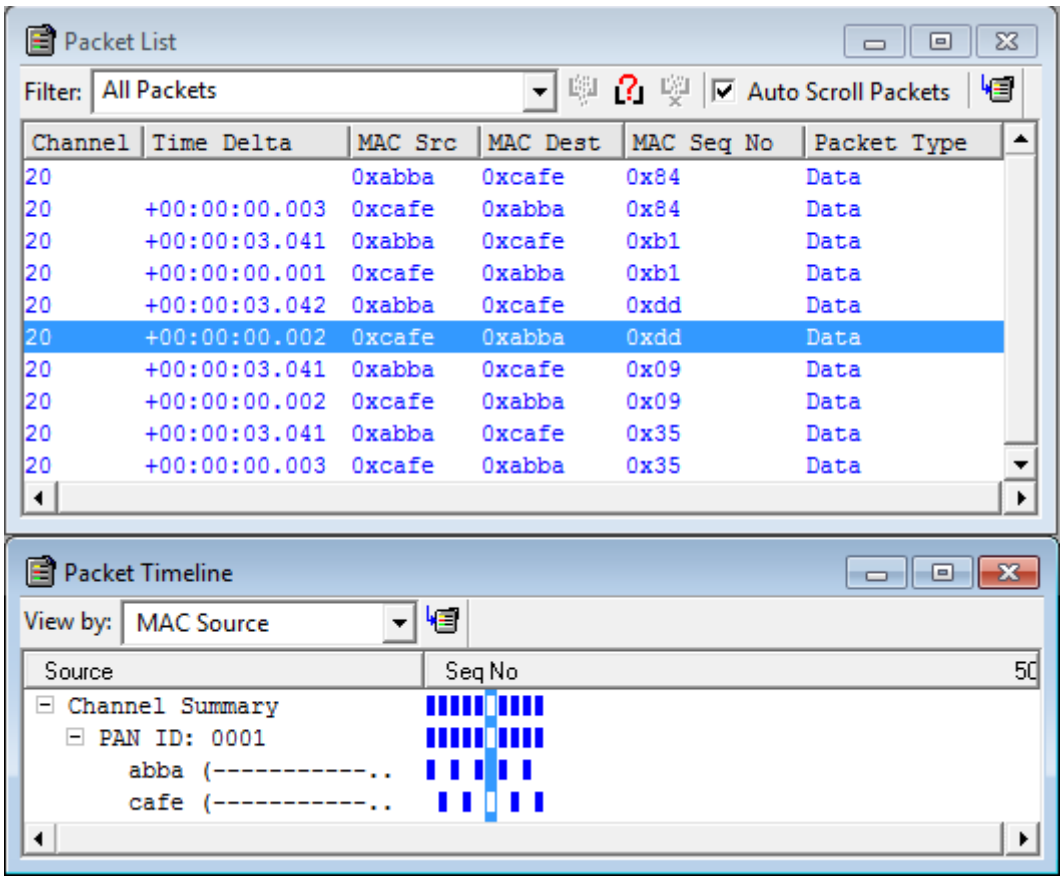


Figur 7-6 DLPDU-spesifikatoren som brukt i evalueringen

Bit nummer fire og fem i DLPDU-spesifikatoren beskriver prioriteten til pakken, og i dette tilfellet (01_2) ser vi at pakken har prioritetstype normal. Bit nummer tre er satt og dette forteller oss at enheten som har sendt ut denne pakken er autentisert i nettverket og at pakkene kan mottas av noden den er adressert til. Bitene null til to beskriver pakketypen, og 111_2 som figuren viser betyr at dette er en datapakke. (Figur 7-3 fremstiller også pakken som en datapakke, men dette er på grunn av den ledende 41_{16} og har ingen sammenheng med pakketypen i WirelessHART). I neste kapittel skal vi se på ACK-ing, og vi forventer at DLPDU-spesifikatoren da beskriver en ACK-pakke. Vi har ikke implementert MIC (Message Integrity Code), og siden CRC sjekkes av programvaren til pakkesnifferen vises den kun som «Correct» i Figur 7-3 og ikke selve verdien som ble sendt.

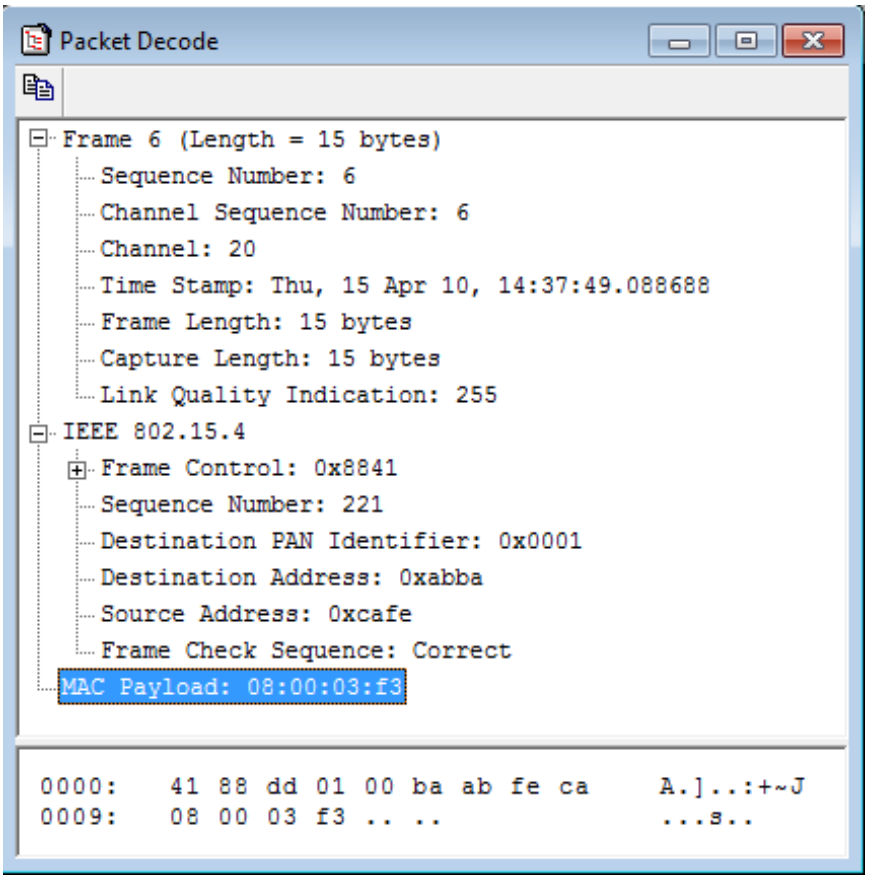
7.3 ACK

Over har vi sett at datatrafikken ut fra en node fungerer som antatt og at pakkene blir bygget opp og sendt ut slik de skal. I dette kapitlet skal vi undersøke hvordan utsending av ACK fungerer, noe som forutsetter to noder som kommuniserer med hverandre. Vi klargjør derfor en node til og gir denne den samme adressen som den første noden er programmert til å sende til. I den nye noden kobler vi ut oppbygging og sending av datapakker. Kommunikasjonen mellom nodene slik programvaren til pakkesnifferen ser den er vist i Figur 7-7.



Figur 7-7 Kommunikasjonen mellom to noder

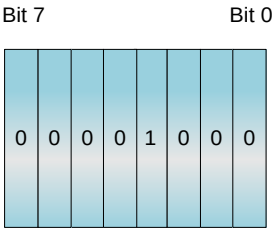
Vi ser at mottakernoden ($CAFE_{16}$) sender ut en ACK med samme sekvensnummer som pakken den nettopp mottok nesten umiddelbart etter mottak. Avsendernoden ($ABBA_{16}$) sender pakker som beskrevet i kapitlene 7.1 og 7.2. Figur 7-8 viser flere detaljer om ACK-pakken.



Figur 7-8 ACK-struktur

Vi husker fra beskrivelsen av ACK DLPDU i kapittel 4.3.4 at nyttelasten skulle bestå av en responskode (en byte) og tidsdifferansen mellom når noden forventet å motta en pakke og når pakken faktisk ble mottatt (to byte). Implementeringskomfigurasjonen vi benytter for evalueringen benytter seg ikke av tidslommer og mottakernoden vet da heller ikke når den burde forvente en innkommende pakke. Vi har, for evalueringens del, heller valgt å legge ved tidsverdien fra en pakke har kommet inn til en ACK skal sendes i retur. Som vi har sett tidligere skal en ACK sendes ut på et gitt tidspunkt i samme tidslomme (for at begge nodene skal ha tid til noe prosessering og til å snu radioene sine fra lytte- til sendemodus og vice versa, og for at avsendernoden skal vite når den kan forvente en ACK i retur). Denne tiden er spesifisert til å være TsTxAckDelay (fra Figur 4-7 på side 54), og kan kontrolleres ved å ta tidspunktet når en hel ramme har kommet inn minus tidspunktet ACK-en sendes ut. Ved å beregne dette tidsintervallet og sammenligne det med tidsvariabelen ser vi hvorvidt vi er i stand til å kontrollere tiden i en node riktig. Vi husker fra Tabell 4-3 at TsTxAckDelay skal være 1000 mikrosekunder (± 100).

Det som i Figur 7-8 er merket blått (og som programvaren tolker som «MAC Payload») er i realiteten DLPDU-spesifikatoren (08_{16}) etterfulgt av ACK-nyttelasten (00 03 $F3_{16}$). Figur 7-9 viser ACK DLPDU-spesifikatoren, og vi ser bit nummer null, én og to definerer pakken til type ACK, og at avsender er innviet i nettverket (bit tre). En forskjell fra data DLPDU-en i Figur 7-6 er at bitene 4 og 5 (pakkens prioritering) ikke er satt. Dette er fordi vi ikke forholder oss til prioritering ved ACK-ing og har heller ikke satt de respektive bitene. Formålet med prioritering av pakker er blant annet å kontrollere flytkontroll, og pakker med prioritet under en satt prioriteringsterskel kan avvises om enheten ikke har nok plass til å ta imot den aktuelle pakken (se kapittel 4.3.4). Ved overføring av en ACK er det ikke behov for midlertidig lagring av pakker i en innkø da ACK-en uansett mottas, behandles, og deretter fjernes innen samme tidslomme som enheten sendte ut datapakken. Dermed er det kun behov for én midlertidig bufferplass til dette.



Figur 7-9 DLPDU-spesifikatoren til ACK-pakken

Etter DLPDU-spesifikatoren kommer nyttelasten i ACK-pakken, og den første her er responskoden som vi for anledningen har satt til å være «Success» (verdi 0), og de to siste bytene er den nevnte tidsverdien vi har beregnet i noden. I motsetning til felt med flere enn én byte i pakkehodet, sendes bytene i *nyttelasten* mest signifikant byte først, og verdien 03 $F3_{16}$ kan leses «rett frem». Når vi oversetter dette til mikrosekunder får vi 1011, som er innenfor kravet om at TsTxAckDelay skal være 1000 ± 100 mikrosekunder. Når ACK-en er klar holder enheten den igjen til det har gått TsTxAckDelay mikrosekunder siden den aktuelle datapakken i sin helhet kom inn. For å se nærmere på hvor mye tid som faktisk benyttes før ACK-en er klar gjorde vi også et forsøk hvor vi returnerte ACK-en så snart den var klar. Resultatet var at noden bruker 493 mikrosekunder fra siste byte har kommet inn på radioen til datalinklaget gir det fysiske laget beskjed om at det er en ACK-pakke som skal sendes ut ved kallet på tjenesteprimitivfunksjonen `data_request(ack_packet)`. På denne tiden skjer følgende:

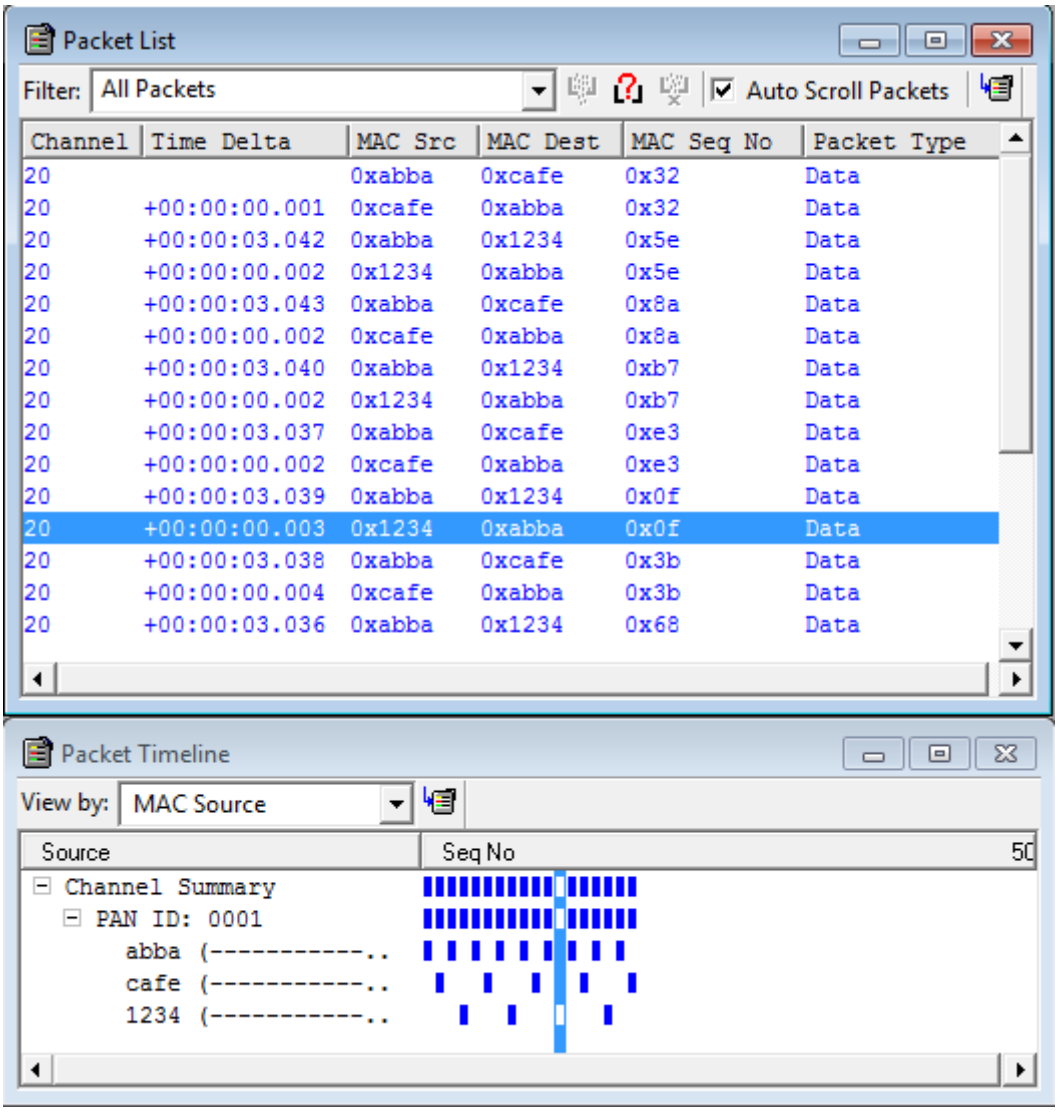
- Pakken lastes fra radioen og inn i minnet
- Pakken konverteres til håndterbar datastruktur
- Pakkens CRC kontrolleres og det sjekkes at den er adressert til noden
- Pakken legges på innkø
- ACK DLPDU blir konstruert
- Radioens retning snus fra mottak- til sendemodus

Det som ikke er implementert er en utvidet prioriteringssjekk før pakken legges på innkøen samt MIC-håndtering (Message Integrity Code), både for den innkommende pakken og for ACK-en som skal genereres. Prioriteringssjekken burde være rask om den implementeres med konstanter som beskriver hvor mange pakker det er plass til for de forskjellige prioriteringene, og det som eventuelt kan ta tid er MIC-håndteringen, men da tiden som er til overs er 518 mikrosekunder (ca 51 % av TsTxAckDelay) antar vi at det er mer enn tid nok.

7.4 Tre noder

For å forsikre oss om at noder evner å kjenne igjen pakker som er adressert til seg selv tilføyer vi en tredje node. Denne gir vi kortadressen 1234_{16}, og vi programmerer noden med adresse $ABBA_{16}$ til å sende annenhver gang til denne nye noden og annenhver gang til den gamle noden ($CAFE_{16}$). Vi ønsker med dette å se om nodene 1234_{16} og $CAFE_{16}$ oppfører seg likt og returnerer ACK kun for pakker som er adressert til dem.

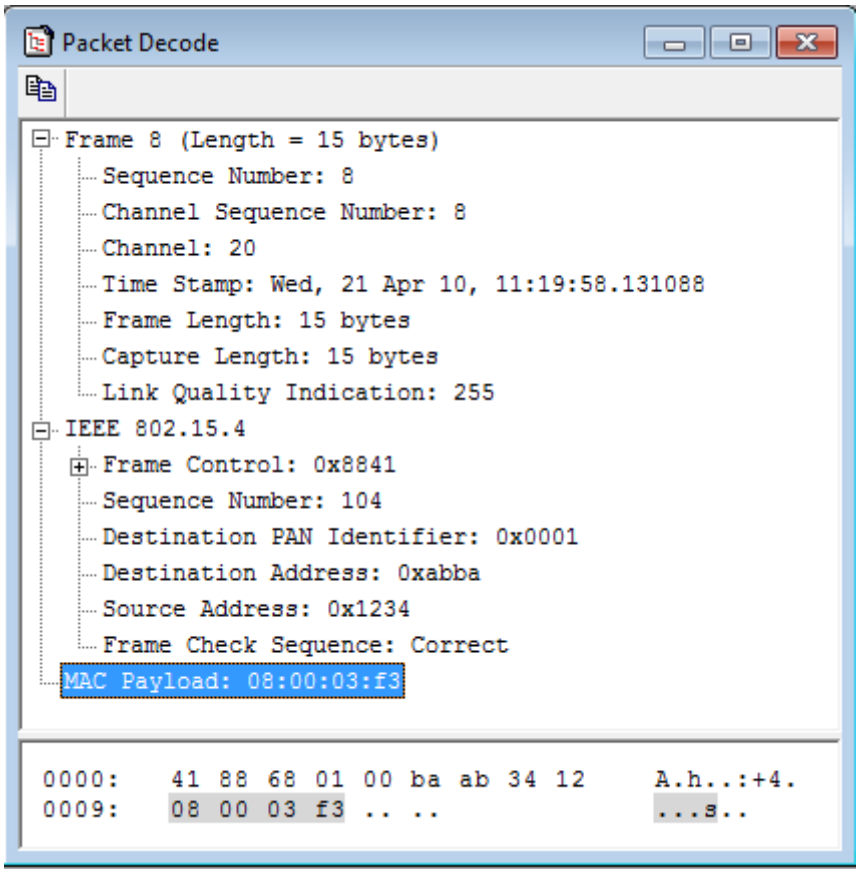
Figur 7-10 viser trafikken etter innføringen av en tredje node (1234_{16}), og om vi sammenligner denne figuren med Figur 7-7 ser vi at tidsintervallene er de samme men at den nye noden har dukket opp i Figur 7-10. Begge figurene viser en «Packet Timeline» i nederste del. Dette viser trafikken i nettverket (beskrevet som «PAN ID: 0001», tilsvarende «Network ID» i WirelessHART, som i vårt tilfelle er hardkodet inn i protokollen som «0001») samt en sekvensiell fremstilling over hvilke noder som sender ut pakker, og vi ser også her at den nye noden har blitt del av nettverket.



Figur 7-10 Trafikken på nett med tre noder i nettverket

Vi kan også sammenligne ACK-pakkene fra de to nodene som noden $ABBA_{16}$ sender data-pakker til (henholdsvis nodene $CAFE_{16}$ og 1234_{16}) ved å sammenligne Figur 7-8 og Figur 7-11. Vi ser at de eneste forskjellene er forskjellige adresser og sekvensnummere.

Med disse undersøkelsene kan vi si at datalinklaget håndterer flere noder i nettverket med tanke på mottak og ACK-ing av data.



Figur 7-11 Strukturen til en ACK fra den nye noden

7.5 Arkitektur og design

Underveis i utviklingen har vi gjort en del arkitektur- og designvalg som nødvendigvis har påvirket implementeringen. Disse valgene har vi analysert og diskutert etter beste evne. Med mer tid til rådighet ville dog noe av analyseringen rundt hvilke løsninger som passer best fått mer oppmerksomhet. Vi har imidlertid gjort oss noen erfaringer underveis, og ut fra dette har vi gjort oss opp noen meninger om hvorvidt våre arkitektur- og designvalg har vært vellykkede.

I det overordnede designet, det vi velger å omtale som arkitekturen, var det første spørsmålet som meldte seg om vi skulle benytte et operativsystem eller ikke. Vi valgte å basere oss på kodebiblioteket AVR2025, som ikke er bygget på et operativsystem men heller er bygget direkte på maskinvaren, da vi anså mulighetene for å optimalisere minne- og energiforbruk for å være større med denne koblingen mot maskinvaren. Så langt vi kan bedømme har vi vært tjent med å velge denne løsningen. Vi hadde tilgang på en ZigBee-implementering ferdig konfigurert for kjøring på Atmel-brikken, med både makefil og en prosjektfil for AVR Studio, og hadde et fungerende ZigBee-nettverk oppe nesten umiddelbart. Å sette seg inn i det relativt

store kodebiblioteket var imidlertid svært tidkrevende, særlig for oss som ikke har noe erfaring med utvikling på små enheter og mikrokontrollere fra før, og det tok lang tid før vi følte oss komfortable i kildekoden. Om dette ville vært enklere med en implementering basert på for eksempel Contiki OS har vi imidlertid vanskelig for å tro, da det sannsynligvis ville vært minst like mye nytt å forholde seg til her. En fordel med å utvikle på et operativsystem som Contiki ville dog vært en bedre kompabilitet med andre brikkesett. Vår implementering er skrevet med tanke på en spesifikk Atmel-brikke, og vil ikke uten videre fungere på andre brikkesett. Implementeringen er imidlertid utført med dette i tankene, og ved å benytte oss av den modulære oppbyggingen i kodebiblioteket AVR2025 bør det ikke kreves store endringer for å programmere vår implementering også på eventuelle andre, WirelessHART-kompatible Atmel-brikker enn de vi har benyttet oss av.

Når det gjelder den utvidede funksjonaliteten et operativsystem kan tilby har vi ikke savnet denne. WirelessHART er en protokoll som ikke har behov for flere samtidige prosesser, og minnehåndtering utføres greit gjennom AVR2025. En ulempe med vår implementering er at AVR2025s timer-funksjonalitet ikke er avbruddsdrevet, men i stedet håndteres via en konstant «polling». Siden vår implementering er basert på dette, innebærer det at vi ikke setter prosessoren i sovemodus der dette ville være naturlig fordi pollingen foregår konstant i ledige øyeblikk (se pseudokode på side 95). I et videre arbeid kan (og bør) dette endres til å ta i bruk mikrokontrollerens omfattende innebygde sovemekanismer, som omtalt i dens dokumentasjon [52].

Valget av AVR2025 førte til at vi la vår egen modularkitektur tett opp til den eksisterende i kodebiblioteket. Vi mener dette er en logisk oppdeling som er gjennomtenkt av Atmel, og som har fungert godt også for oss. Derfor mener vi arkitekturen vil gjøre det greit å bygge videre på vår implementering, og i tillegg er den gjenkjennelig for enhver som måtte være kjent med AVR2025-kodebiblioteket fra før.

På det mer konkret implementeringstekniske gjorde vi en ganske stor omveltning i programflyten i forhold til AVR2025 ved at vi gikk bort fra designet med tjenesteprimitiver implementert som datastrukturer som ble sendt mellom lagene via hendelseskøer. I stedet valgte vi å implementere tjenesteprimitivene som rene funksjonskall som ble kalt direkte fra et lag til et annet. Vi mente at dette lot seg gjøre grunnet WirelessHART-protokollens sekvensielle natur, og at behovet for køing av hendelser ikke var nødvendig. Dette mener vi var et fornuftig valg, da det forenklet implementeringen vesentlig. Vi kunne forkaste flere køer og datastrukturer fra kildekoden, og reduserte dermed kodestørrelsen og kompleksisteten

vesentlig uten at det har gått ut over funksjonaliteten. Andre konkrete designvalg er diskutert i implementeringskapitlet, og er ikke evaluert ut over den funksjonelle testingen tidligere i dette kapitlet. Vi har allerede sett at de delene vi har implementert holder seg innenfor standardens krav til tidsforbruk, og ut over dette overlater vi til senere arbeider å vurdere hvorvidt de ulike valg og algoritmer bør og kan effektiviseres. Som nevnt tidligere har vi valgt å fokusere mer på funksjonalitet enn effektivitet i denne fasen av utviklingen.

8 Konklusjon og fremtidig arbeid

I tillegg til en nødvendig innledende forklaring av trådløse sensornettverk generelt og IEEE 802.15.4-standarden spesielt, har vi nå tatt for oss både en grundig gjennomgang av de to nederste lagene i protokollstakken WirelessHART og gått gjennom vår egen begynnende implementering av disse, samt utført tester på denne implementeringen. Vi skal her først oppsummere hva vi har kommet frem til, før vi til slutt foreslår hvilke områder det burde jobbes videre med ved en fortsettelse av implementeringen mot en fullstendig WirelessHART-protokoll.

8.1 Konklusjon

Ved vår implementering har vi fått på plass flere grunnleggende elementer, og gjennom dette arbeidet mener vi at en fullstendig implementering av WirelessHART-protokollen på brikkene fra Atmel (som primært er utviklet for IEEE 802.15.4 og ZigBee) er gjennomførbart, og oppfordrer derfor til videre utvikling av vår implementering.

Vi har gjort alle de nødvendige justeringene på det fysiske laget fra IEEE 802.15.4 for at det skal stemme overens med WirelessHART. På datalinklaget har vi implementert så godt som et fullstendig LLC-sublag (Logical Link Control), med unntak av oppbyggingen av noen DLPDU-pakketyper og sikkerhetshåndteringen. På MAC-sublaget er linkbehandleren med modulene for overføring og mottak (XMIT- og RECV-motorene) implementert. Den kanskje mest sentrale delen av MAC-sublaget er linkfordeleren, og det aller meste av denne har vi implementert. På tidspunktet hvor vi foretok testing og evaluering var linkfordeleren koblet ut, da noe implementering av vår egen listehåndterer gjenstod og førte til at den ikke fungerte som tenkt. Vi hadde tidligere en fungerende, men ikke helt komplett, versjon av linkfordeleren, og vi mener at logikken skal være på plass.

Gjennom arbeidet med denne rapporten og implementeringen har vi tilordnet oss inngående kunnskap om WirelessHART og IEEE 802.15.4. WirelessHART er en ny standard som foreløpig få har kunnskap om, men som antas å etablere seg blant de ledende alternativene for trådløse sensornettverk. Vi har ved denne rapporten forsøkt å formidle kunnskapen vi selv har tilegnet oss og håper vi har lykkes med det.

8.2 Fremtidig arbeid

Vi mener å ha utviklet det grunnleggende i tråd med WirelessHART-spesifikasjonen, og basert på dette, kombinert med evalueringen av kravene som stilles til radioen (som vi gikk gjennom i kapittel 4.2.4, og evaluerte i kapittel 5.1.1), mener vi at det er fullt mulig å implementere en fullstendig WirelessHART-protokoll på Atmel-nodene vi har arbeidet på. Vi har listet opp hva vi mener er de viktigste neste punktene å få på plass i tillegg til grunnlaget som er lagt:

- *Tidssynkronisering* – vår implementering er laget med tanke på å etter hvert dekke tidssynkronisering. Noe er derfor allerede klart for en fullføring av implementeringen av tidssynkronisering. Her handler det om å forholde seg til «tidsvinduet» en mottaker har til rådighet for å ta imot en pakke, tiden fra den kan starte lytting til den skal slutte å lytte etter innkommende pakke (gitt ved tidsvariabelen TsRxWait i Figur 4-7), samt dedikere noen noder som tidskilder og synkronisere nettverksklokken etter disse.
- *Kanalhopping* – ved kanalhopping må nodene forholde seg til hvilke kanaler som skal benyttes til enhver tid. Når denne funksjonaliteten er på plass vil bruken være lik som i vår implementering hvor man sender med kanal som parameter i PHY tjenesteprimitiven ENABLE.request. Om man implementerer en algoritme for å vite hvilken kanal som gjelder for den aktuelle overføringen før dette kallet utføres, er selve «hoppingen» mellom kanalene allerede implementert.
- *Svartelisting* – svartelisting refererer til muligheten for å koble ut spesifikke kanaler slik at disse ikke blir benyttet i nettverket. Dette punktet er relatert til punktet over (kanalhopping) ved at en håndtering av flere kanaler må være på plass i implementeringen.
- *Sikkerhet* – For industrien er pålitelige sikkerhetsmekanismer en viktig faktor når de skal ta i bruk trådløs teknologi. Vi har ikke sett på sikkerhetsmekanismene i WirelessHART, men ut fra hva vi vet om sikkerhet innen trådløs kommunikasjon tror vi implementeringen av sikkerhetsmekanismene er overkommelige. Hart Communication Foundation forsøker å benytte åpne standarder hvor det er mulig, og sikkerhet er et slikt eksempel. Det er derfor rimelig å anta at store deler av logikken og algoritmene som skal til for å implementere en sikkerhetsadministrator er tilgjengelig i aktive miljøer innen fagfeltet.

- *Nettverkslaget* – i kapittel 4.4.1 tok vi for oss noen ytterst få egenskaper ved nettverkslaget i WirelessHART og i vår implementering har vi omgått dette laget. Lagene over nettverkslaget er lag som også benyttes i HART-protokollen og nettverkslaget er således det siste laget fra WirelessHART-spesifikasjonen som behøves når det fysiske laget og datalinklaget herfra er implementert. (For at nodene og nettverket som sådan skal fungere må man selvsagt også utvikle lagene over nettverkslaget, men da er det den opprinnelige HART-standarden man skal forholde seg til).
- *Administrasjonstjenester* – av tjenesteprimitiver er det i vår implementering kun implementert tjenester rundt sending og mottak av pakker, og ikke tjenester som dreier seg om nettverks- og enhetskonfigurering, som er administrasjonstjenestenes ansvar. Disse har vi omgått helt, både for det fysiske laget og for datalinklaget, fordi vi har sett på dem som lite relevante på vårt stadie av implementeringen. Disse tjenestene administrerer enheter slik at nettverket fungerer som det skal eller de benyttes til statistikk og diagnostikk over nettverket og de er typisk initiert av nettverksmanager. Vi ser det som naturlig at disse implementeres samtidig med at nettverkslaget (punktet over) implementeres.

9 Ordliste

ACK (ACK DLPDU)
Bekreftelse på en mottatt DLPDU som var spesifikt adressert og ikke kringkastet. ACK er i WirelessHART-sammenheng en type DLPDU som inneholder informasjon om hvilken DLPDU som bekreftes mottatt, hvorvidt mottaket var vellykket eller ikke, og tidsavvik.

Ad-Hoc-nettverk
Nettverk som er selvorganiserende. Slike nettverk håndterer at noder kobler seg av og på og nettverket omorganiserer seg selv deretter.

ASN – Absolute Slot Number
ASN er en teller som holder rede på antall tidslommer som har funnet sted siden nettverkets oppstart. Telleren kan ikke tilbakestilles og økes med én for hver tidslomme som går. ASN er dermed til enhver tid aktuell tidslomme.

AVR
Navn på en «familie» mikrokontrollere fra Atmel.

CA – Collision Avoidance
Brukes sammen med CSMA for deling av overføringsmediet mellom flere brukere. CA er en teknikk hvor noder avverger kollisjoner ved å holde tilbake sending om noden har oppdaget at det er trafikk på overføringsmediet. Overføring forsøkes så ved senere anledning.

CCA – Clear Channel Assesment
Teknikk for å forsikre seg om at kanalen er klar. Enheten lytter til kanalen før sending, og om det oppdages signal kan overføringen utsettes.

CCM* – Counter with CBC-MAC (endret)
Metode for å kryptere og autentisere data. CCM* er en endring av CCM slik at det er mulig å benytte kryptering alene eller autentisering alene.

CD – Collision Detection
Brukes sammen med CSMA for deling av overføringsmediet mellom flere brukere. CD er en teknikk hvor nodene oppdager kollisjon på overføringsmediet og kan avbryte videre sending. (For trådløs overføring kan ikke CD benyttes da man ikke kan oppdage kollisjoner – her kan CA erstatte CD).

CRC – Cyclic Redundancy Check
En verdi beregnet over en mengde data. Verdien beregnes på begge sider av en overføring og er ment til å oppdage små feil i overført data grunnet for eksempel støy.

CSMA – Carrier Sense Multiple Access

Protokoll som benyttes for tilgang til delt overføringsmedium. Enheter som ønsker å sende data lytter først til kanalen for å avgjøre om den er i bruk eller ikke før man starter sending.

dBm

Forkortelse for desibel (dB) i forhold til en milliwatt (1 mW). Eksempler: 0 dBm = 1 mW. 10 dBm = 10 mW. 20 dBm = 100 mW. 30 dBm = 1 W.

DLL – Data Link Layer

Datalinklaget er ansvarlig for feilfri overføring av data mellom to enheter, og det er dette laget som definerer meldingsstrukturen samt strategien for håndtering av feil. (Se også MAC og LLC).

DLPDU – Data Link layer Protocol Data Unit

Datapakken som håndteres av datalinklaget. Pakken kan inneholde en datapakke fra laget over som nyttelast, men kan også bestå av informasjon som kun er til bruk innad i datalinklaget (For eksempel ACK DLPDU). Utover eventuell nyttelast inneholder pakken alltid informasjon til bruk internt i datalinklaget i pakkehodet og –halen.

EIRP – Equivalent Isotropic Radiated Power

Effekten som en (teoretisk) isotropisk antenne (antenne som gir samme effekt i alle retninger) må gi fra seg for å oppnå samme effekt som målt som høyeste effekt i en reell antenne (ikke nødvendigvis isotropisk).

FFD – Full Function Device

Én av to typer enheter definert i IEEE 802.15.4-standarden. FFD inneholder hele settet med MAC-tjenester og kan derfor fungere både som vanlig node i et nettverk og som nettverkskoordinator.

Grafruting

Rutingteknikk hvor en rettet graf representerer veien fra en node til en bestemt mottaker. Grafen kan (og bør) være redundant med alternative veier. Når man i WirelessHART benytter grafruting leser datalinklaget av grafens ID og ser på informasjon som datalinklaget har lagret om mulige naboer som representerer neste hopp i den spesifikke grafen.

HART – Highway Addressable Remote Transducer protocol

Trådbasert protokoll for automatisering i industrien. HART er en enkel og robust protokoll og kan kommunisere over 4-20 mA analoge sløyfer. Protokollen forvaltes av HCF.

HCF – Hart Communication Foundation

Forening som forvalter den trådbaserte protokollen HART og den trådløse WirelessHART. Foreningen er nonprofitt og finansieres hovedsakelig av medlemsbedrifter som har interesse i videreutviklingen av de to protokollene.

IEC – The International Electrotechnical Commission
Internasjonal organisasjon som skriver og publiserer standarder for teknologier som er forbundet med elektronikk og elektrisitet.

IEEE – Institute of Electrical and Electronics Engineers
Internasjonal nonprofitt-organisasjon som arbeider med å videreutvikle og forbedre teknologi knyttet til elektrisitet.

IEEE 802.15.4 (LR-WPAN)
WPAN utviklet med tanke på lav datarate og lavt energiforbruk. Er utarbeidet av IEEE. Det finnes to varianter: Den opprinnelige IEEE 802.15.4-2003 og videreutviklingen av denne, IEEE 802.15.4-2006.

ISA – The International Society of Automation
Nonprofitt-organisasjon bestående av representanter fra industrien, utdanningsinstitusjoner og næringsliv. ISA arbeider med automatisering og instrumentering i industrien.

ISA 100.11a
Spesifikasjon for IEEE 802.15.4-nettverk som baserer seg på IEEE 802.15.4-2006 med et modifisert MAC-lag. Standarden ble ISA-ratifisert september 2009 og forsøker å konvergere flere trådløse industristandarder. Utviklet og under forvaltning av ISA.

Kanal
Avgrenset frekvensområde som benyttes for overføring av data. I WirelessHART er hver kanal 5 MHz bred og kan være en av 15 kanaler mellom 2 405 MHz og 2 475 MHz.

Kanalhopping
Teknikk for hyppig endring av sende-/mottakerfrekvens. Effektivt mot interferens og signaldegradering ved multiple stier.

Kilderuting
Rutingteknikk hvor hele ruten frem til adressaten ligger i pakken. For hvert hopp i nettverket leser noden av neste hopp og fjerner dette fra pakken før pakken sendes videre. Gjentas til pakken har kommet frem til destinasjonen.

Link
Nødvendig kommunikasjonsinformasjon for å overføre en pakke mellom to nabonoder. Informasjonene består av adressene (avsender og mottaker), tidslomme, kanaloffset, retning (sende eller motta), hvorvidt linken er dedikert (mellom to noder) eller delt, og type link.

Linkfordeler
Modul i MAC-sublaget som har som oppgave å avgjøre og planlegge hvilken link som skal behandles til hvilken tid. Linkfordeleren skal til enhver tid ha oversikt over hvilken link som er den neste som skal betjenes og kjøres blant annet etter hver betjente tidslomme og ved endringer utstedt av nettverksadministrator.

LLC – Logical Link Control
Øverste av to sublag innad i datalinklaget. LLC-laget håndterer konstruksjon av DLPDU-er, feilhåndtering og flytkontroll.

MAC - Medium Access Control
Nederste av to sublag innad i datalinklaget. MAC-laget håndterer tilgang til kommunikasjonskanalen. MAC-lagets hovedoppgave er å sørge for feilfri overføring av DLPDU-er mellom to nabonoder.

MIC – Message Integrity Code
Kode som benyttes for autentisering og kryptering av pakker. Koden genereres vanligvis av en offentlig og en hemmelig nøkkel samt innholdet som skal sendes. (I WirelessHART benyttes MIC kun for autentisering).

Nettverksadministrator
Programvare (anbefalt plassert i gateway) som er ansvarlig for konfigurering av nettverket, administrasjon av kommunikasjon mellom nettverksenheter, rutingtabeller, samt overvåking av nettverkets tilstand ved å jevnlig innhente statistikk over enheter og trafikken mellom dem. Det er nettverksadministrator som distribuerer enheters korte adresser og gir tilgang til nettverket.

NPDU – Network layer Protocol Data Unit
Datapakke som håndteres av nettverkslaget. NPDU-en kan videreformidle datapakker fra høyere lag.

PAN – Personal Area Network
Nettverk som skal fungere i det nærmeste arbeidsområdet til en person.

PAS – Publicly Available Specification
Dokument som beskriver forslag til IEC-standard og som må gjennom en godkjenningsprosess før forslaget eventuelt blir godkjent.

PHY – Physical Layer
Det fysiske laget. Dette laget utgjør den fysiske kontakten med overføringsmediet. Det er dette laget som står for overføringen av bitstrømmen på overføringsmediet, og det er på dette laget signaleringsparametrene for overføringssignalet blir satt.

PIB – PAN Information Base
En samling nettverksrelatert informasjon som det aktuelle laget behøver for å fungere som tiltenkt. Både det fysiske laget og datalinklaget har sin egen PIB.

PPDU – Physical layer Protocol Data Unit
Datapakken som håndteres av det fysiske laget. Pakken inneholder datapakken fra laget over som nyttelast samt informasjon til bruk internt i det fysiske laget i pakkehode og –hale.

RFD – Reduced Function Device
En av to typer enheter definert i IEEE 802.15.4-standarden. RFD er utviklet med tanke på å holde resursforbruket nede og inneholder et minimum av tjenester. RFD-er kan kun fungere som en enkel node i et nettverk og kan blant annet ikke videresende pakker.

Signaldegradering ved multiple stier (Multipath fading)
Fenomen som oppstår når radiosignaler reflekteres av objekter, som igjen fører til at de samme signalene blir mottatt flere ganger fra forskjellige stier og dermed skaper interferens.

Superramme
En samling tidslommer. En superramme avgrenses ved antall tidslommer i superrammen og repeteres umiddelbart etter seg selv. En superramme (i WirelessHART) kan være aktiv eller inaktiv, og det kan gå flere superrammer parallelt i nettverket. Det må alltid være minst én aktiv superramme i nettverket. Superrammer administreres av nettverksadministrator.

Svartelisting
Svartelisting refererer til muligheten til å forby kommunikasjon på noen kanaler. Kommunikasjonen innen nettverket vil da foregå på de resterende kanalene. Kan for eksempel benyttes i nettverk hvor visse frekvenser er utsatt for støy fra annet utstyr og overføring sannsynlig hvis vil mislykkes.

TDMA – Time Division Multiple Access
Protokoll for å styre tilgangen til et delt medium i den hensikt å unngå interferens og kollisjoner. Kanaltilgangen deles inn i tidslommer av fast lengde som grupperes i rammer av fast antall tidslommer (se superramme). Bruken av TDMA (i WirelessHART-varianten) bidrar til et deterministisk nettverk, samt at noder kan skru av radioen når de ikke har tildelte tidslommer for å spare strøm.

Tidslomme
Tidsperiode på 10 millisekunder hvor kommunikasjon mellom noder kan skje. En overføring må fullføres i løpet av tidslommen, inkludert bekreftelse på overføringen ved retur av en eventuell ACK DLPDU.

Tjenesteprimitiv
Overordnet spesifikasjon av en tjeneste som tilbys mellom lagene. Et tjenesteprimitiv kan være en av de fire typene request (sendes fra et lag til laget under for å initiere en tjeneste), confirm (sendes fra underliggende lag for å formidle resultatet av tidligere utført tjeneste som følge av en request), indication (sendes fra underliggende lag til laget over for å indikere en hendelse som er av betydning for laget over) eller response (sendes til underliggende lag som avslutning av en prosedyre satt i gang som følge av en tidligere sendt indicate).

TPDU – Transport layer Protocol Data Unit
Datapakke som håndteres av transportlaget.

WirelessHART
Den trådløse adapteringen av den trådbaserte protokollen HART. Baserer seg på IEEE 802.15.4-2006 og er utviklet for å imøtekomme behovet for en robust og sikker protokoll for trådløse sensornettverk i industrien. WirelessHART Forvaltes av HCF.

WPAN – Wireless Personal Area Network
Trådløst PAN. Typisk rekkevidde er 10 meter i alle retninger. WPAN er standardisert av IEEE og omfattes av IEEE 802.15-standardene.

ZigBee
Spesifikasjon for IEEE 802.15.4-basert nettverk rettet mot hjemme-automatisering, overvåking og kontroll. Spesifikasjonen kom i 2004. Baserer seg på IEEE 802.15.4-2003.

ZigBee Pro
Videreutvikling av ZigBee for å dekke behovene i industrien. Baserer seg på IEEE 802.15.4-2006.

10 Appendiks A – WirelessHART-datalinklagets «local management»-tjenesteprimitiver

Under vises en oversikt over «local management»-tjenesteprimitivene for datalinklaget. Tabellen er hentet fra WirelessHART-standarden, side 33 til 35.

Table 1 – Local Device Management Commands

Service	Data	Description
RESET		Initializes the Data-Link Layer
DISCONNECT		Disconnect from the network, cease communications
RE_JOIN		Disconnect from the network, rejoin the network, purging all MAC queues and clearing all MAC tables
WRITE_SUPERFRAME		Creates a new superframe
	Unsigned-8 superframeId	
	Unsigned-16 nSlots	Length of superframe
	Boolean active	Activates (TRUE) or de-activates (FALSE) the superframe
DELETE_SUPERFRAME		Deletes an existing scheduling superframe and any associated links
	superframeId	
ADD_LINK		Adds a new link to another device, possibly updating the neighbor and connection tables in the process
	Unsigned-8 linkHandle	handle of created link record, if any
	Unsigned-8 superframeId	
	Unsigned-16 nodeAddress	Address of neighbor device
	Unsigned-16 slot	Slot in the superframe to use by this link
	Unsigned-8 channelOffset	Offset of logical channel relative to base channel for this slot
	linkOptions	bitmap: {Transmit=b001, Receive=b010, Shared=b100}
	linkType	One of {NORMAL, JOIN, DISCOVERY}
DELETE_LINK		Deletes an existing link, possibly updating the neighbor and connection tables in the process
	linkHandle	
ADD_CONNECTION		Adds a new connection to a device via a specified graph
	Unsigned-8 connectionHandle	Handle for this connection
	Unsigned-16 graphId	graphId for the connection
	Unsigned-16 nodeNickname	address of device being connected via the specified graph

Service	Data	Description
DELETE_CONNECTION		Deletes an existing connection
	Unsigned-8 connectionHandle	Handle of an existing connection
READ_NETWORKID		Reads the ID of the network the device belongs to
	Unsigned-16 NetworkID	
WRITE_NETWORKID		Writes the ID of the network the device belongs to
	Unsigned-16 NetworkID	
WRITE_NETWORK_KEY		This command allows the Network Manager to write the network key on a Network Device. Keys should be protected from pilfering (e.g., by encryption)
	Unsigned-128 networkKey	
	Unsigned-48 SlotNumber	Execution time for command (ASN) (0 means execute immediately)
READ_TIMEOUT _PERIODS		
	Time keepAliveInterval	Interval during which a node shall successfully communicate with each linked neighbor. Any DLPDU received from the neighbor resets the Keep-Alive timer for that neighbor
	Time pathFailInterval	Interval of unsuccessful communication with a given neighbor, indicating a path failure
	Time advertiseInterval	Time period specifying the transmission of Advertise DLPDUs
	Time discoveryInterval	Time period specifying the interval bounding the random transmission of Advertise DLPDUs on Discovery links
WRITE_TIMEOUT _PERIOD		Write the indicated time period value.
	Unsigned-8 timerCode	One of { Keep-Alive; Path-Failure; Advertise; or Discovery }
	Time timerPeriodValue	
READ_CAPACITIES		
	Unsigned-16 maxSuperframes	
	Unsigned-16 maxLinks	
	Unsigned-16 maxNeighbors	
	Unsigned-16 maxPktBuffers	
READ_PRIORITY _THRESHOLD		
	Unsigned-4 priorityThreshold	Specifies the lowest priority DLPDU to be accepted from another device
WRITE_PRIORITY _THRESHOLD		
	Unsigned-4 priorityThreshold	
READ_JOIN_PRIORITY		Indicates what join priority the device shall advertise Lower number indicates a better choice for joining
	Unsigned-4 joinPriority	
WRITE_JOIN_PRIORITY		
	Unsigned-4 joinPriority	
READ_PROMISCUOUS _MODE		Indicates whether the sublayer is in “receive all” mode. TRUE indicates the sublayer accepts all PDUs received from the Physical Layer
	Boolean promiscuousMode	

Service	Data	Description
WRITE_PROMISCUOUS _MODE		
	Boolean promiscuousMode	
READ_MAX_BACK_OFF _EXPONENT		The maximum value that can be assumed for the back-off exponent used in shared slots. Valid values are { 4, 5, 6, 7 } MaxBackoffExponent defaults to 4
	Unsigned-4 MaxBackoffExponent	
WRITE_MAX_BACK _OFF_EXPONENT		
	Unsigned-4 MaxBackoffExponent	
ADD_CONNECTION		Adds a new connection to a device with specified graphs
	Unsigned-16 connectionhandle	Handle for this connection
	Unsigned-16 grasphId	graphID for the connection
	Unsigned-16 node Nickname	Address of device being connected via the specified graph
DELETE_CONNECTION		Deletes an existing connections
	Unsigned-16 connectionhandlet	Handle of an existing connection

11 Referanser

1. Dust Networks. *Dust Networks Announces Production Shipment of Embedded WirelessHART Sensor Networking Solution*. 2008. http://www.dustnetworks.com/node/33 (lest 19.01.2010).
2. Wireless Industrial Technology Konsortium - WiTECK. *WiTECK: our work*. 2009. http://www.witeck.org/our_work/index.html (lest 19.01.2010).
3. Texas Instruments. *CC2420: 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver*. 2007. http://focus.ti.com/lit/ds/symlink/cc2420.pdf.
4. Atmel. *AT86RF230: Low Power 2.4 GHz Transceiver for ZigBee, IEEE 802.15.4, 6LoWPAN, RF4CE and ISM Applications*. 2009. http://www.atmel.com/dyn/resources/prod_documents/doc5131.pdf.
5. ControlGlobal.com. *The WirelessHART Ecosystem - A Scorecard for End Users to Enjoy*. 2009. http://community.controlglobal.com/content/wirelesshart-ecosystem-scorecard-end-users-enjoy (lest 17.01.2010).
6. NIVIS. *Wireless sensors and network control*. 2008. http://www.nivis.com/technology/WirelessHART.aspx (lest 12.02.10).
7. Dust Networks. *DN2510: 2.4 GHz Mote-on-Chip*. 2007. http://www.dustnetworks.com/sites/default/files/015-0018IA-510(I)rev1DN2510ProductBrief.pdf.
8. Gutiérrez, J.A., E.H. Callaway, og R.L. Barrett, *Low-rate wireless personal area networks : enabling wireless sensors with IEEE 802.15.4*. IEEE Standards Wireless Networks Series. 2003, New York: IEEE.
9. Hanssen, L. og J. Gakkestad, *Det moderne Forsvaret bygger på elektronikk.* FFI-fokus, 2006(4): s. 2-8.
10. Culler, D., D. Estrin, og M. Srivastava, *Guest Editors' Introduction: Overview of Sensor Networks.* Computer, 2004. **37**(8): s. 41-49.
11. Kahn, J.M., R.H. Katz, og K.S.J. Pister, *Emerging challenges: Mobile networking for "Smart Dust"*. Journal of Communications and Networks, 2000. **2**(3): s. 188-196.
12. Hanssen, L., *Wireless Sensor Networks: Trådløse sensornettverk*. 2006, Kjeller: FFI.
13. Flammini, A., P. Ferrari, D. Marioli, E. Sisinni, og A. Taroni. *Sensor networks for industrial applications*. i *Advances in Sensors and Interface, 2007. IWASI 2007. 2nd International Workshop on*. 2007.
14. Petersen, S., P. Doyle, S. Vatland, C.S. Aasland, T.M. Andersen, og S. Dag. *Requirements, drivers and analysis of wireless sensor network solutions for the Oil & Gas industry*. i *Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on*. 2007.
15. ISO/IEC. *Information Technology - Open Systems Interconnection - Basic Reference Model: The Basic Model*. 1994.
16. ANSI, *Information processing systems - Local area networks - Part 2: logical link control.* ISO Std 8802-2: 1998; IEEE Std 802.2-1998, 1989.
17. Tanenbaum, A.S., *Computer networks*. 4th ed. 2003, Upper Saddle River, NJ: Prentice Hall PTR.
18. IEEE. *IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 15.4: Wireless Medium Access Control (MAC) and*

Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs). 2006.
19. IEEE, *IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, i *IEEE Std 802.11-2007 (Revision of IEEE Std 802.11-1999)*. 2007, IEEE Computer Society.
20. IEC. *Industrial communication networks: Fieldbus specifications : WirelessHart communication network and communication profile : IEC/PAS 62591, Ed. 1.0 (22.01.2009)*. 2009. International Electrotechnical Commission.
21. Held, G., *Data over wireless networks : Bluetooth, WAP, and wireless LANS*. 2001, New York, NY: McGraw-Hill.
22. Misic, J. og V.B. Misic, *Wireless personal area networks : performance, interconnections and security with IEEE 802.15.4*. 2007, Chichester, West Sussex, England ; Hoboken, NJ, USA: Wiley.
23. IEEE. *IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)*. 2003.
24. ZigBee Alliance. *ZigBee Specification*. 2008.
25. Lennvall, T., S. Svensson, og F. Hekland. *A comparison of WirelessHART and ZigBee for industrial applications*. i *Factory Communication Systems, 2008. WFCS 2008. IEEE International Workshop on*. 2008.
26. ZigBee Alliance. *ZigBee-PRO Stack Profile*. 2008.
27. Petersen, S. og S. Carlsen. *Performance evaluation of WirelessHART for factory automation*. i *Emerging Technologies & Factory Automation, 2009. ETFA 2009. IEEE Conference on*. 2009.
28. ISA. *ISA wireless standard OK'd*. 2009. http://www.isa.org/Content/ContentGroups/News/2009/September39/ISA_wireless_standard_OKd.htm (lest 24.02.2010).
29. Xuan, Z., W. Min, W. Ping, og K. Yeon. *Research and implementation of security mechanism in ISA100.11a networks*. i *Electronic Measurement & Instruments, 2009. ICEMI '09. 9th International Conference on*. 2009.
30. Al Agha, K., M.H. Bertin, T. Dang, A. Guitton, P. Minet, T. Val, og J.B. Viollet, *Which Wireless Technology for Industrial Wireless Sensor Networks? The Development of OCARI Technology*. Industrial Electronics, IEEE Transactions on, 2009. **56**(10): s. 4266-4278.
31. Biasi, M.D., C. Snickars, K. Landernas, og A. Isaksson, *Simulation of Process Control with WirelessHART Networks Subject to Clock Drift*, i *Proceedings of the 2008 32nd Annual IEEE International Computer Software and Applications Conference*. 2008, IEEE Computer Society. s. 1355-1360.
32. Rios, M., *Unplugged - Developing standards for wireless automation*. Pharmaceutical Technology, 2008. **32**(5): s. 40.
33. ISA. *ISA wireless collaborates with HART*. 2007. http://www.isa.org/InTechTemplate.cfm?Section=Industry_News&template=/ContentManagement/ContentDisplay.cfm&ContentID=64763 (lest 14.11.2009).
34. Bond, A. *ANSI ISA 100.11a Approval Stalled* i *ControlGlobal.com*. 2010. http://www.controlglobal.com/industrynews/2010/015.html (lest 25.02.2010).

35. HART Communication Foundation. *WirelessHART Approved by IEC as First International Standard for Wireless Communication in Process Automation*. 2010. http://hartcomm.org/hcf/news/whats_new/WirelessHART_Approved_by_IEC.html (lest 28.04.2010).
36. HART Communication Foundation. *NAMUR Confirms WirelessHART for Process Applications*. 2009. http://www.hartcomm.org/hcf/news/pr2009/namur_nov2009.html (lest 28.02.2010).
37. HART Communication Foundation. *HART Communication Protocol and Foundation*. 2009. http://hartcomm.org/ (lest 16.02.2010).
38. Kim, A.N., F. Hekland, S. Petersen, og P. Doyle. *When HART goes wireless: Understanding and implementing the WirelessHART standard*. i *Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on*. 2008.
39. Internet Engineering Task Force. *RFC3610: Counter with CBC-MAC (CCM)*. 2003. Network Working Group, IETF. http://tools.ietf.org/html/rfc3610.
40. IEC. *Industrial communication networks - Fieldbus specifications - Part 5-20: Application layer service definition - Type 20 elements* 2007.
41. IEC. *Industrial communication networks - Fieldbus specifications - Part 6-20: Application layer protocol specification - Type 20 elements* 2007.
42. Atmel. *AVR JTAG ICE User Guide*. 2001. http://www.atmel.com/dyn/resources/prod_documents/doc2475.pdf.
43. SourceForge.net. *AVR plug-in for Eclipse IDE*. 2010. http://avr-eclipse.sourceforge.net/wiki/index.php/The_AVR_Eclipse_Plugin (lest 09.03.2010).
44. Daintree Networks, *Sensor Network Analyzer Quick Start Guide*. 2007.
45. Atmel. *Atmel homepage*. 2009. http://www.atmel.com/ (lest 17.11.2009).
46. Atmel, *AVR2025: IEEE 802.15.4 MAC Software Package - User Guide*. 2009.
47. Gustafsson, D., *WirelessHART- Implementation and Evaluation on Wireless Sensors*. 2009, Electrical Engineering, Kungliga Tekniska högskolan: Stockholm.
48. Biasi, M.D., *Implementation of a WirelessHART simulator and its use in studying packet loss compenstation in networked control*. 2008, Electrical Engineering, Kungliga Tekniska högskolan: Stockholm.
49. Jianping, S., H. Song, A.K. Mok, C. Deji, M. Lucas, og M. Nixon. *WirelessHART: Applying Wireless Technology in Real-Time Industrial Process Control*. i *Real-Time and Embedded Technology and Applications Symposium, 2008. RTAS '08. IEEE*. 2008.
50. Durvy, M., J. Abeillé, P. Wetterwald, C. O'Flynn, B. Leverett, E. Gnoske, M. Vidales, G. Mulligan, N. Tsiftes, N. Finne, og A. Dunkels, *Making sensor networks IPv6 ready*, i *Proceedings of the 6th ACM conference on Embedded network sensor systems*. 2008, ACM: Raleigh, NC, USA. s. 421-422.
51. Atmel. *AVR053: Calibration of the internal RC oscillator*. 2006. http://www.atmel.com/dyn/resources/prod_documents/doc2555.pdf.
52. Atmel. *ATmega 1284P Preliminary*. 2009. http://www.atmel.com/dyn/resources/prod_documents/doc8059.pdf.